Consultative Committee for Space Data Systems

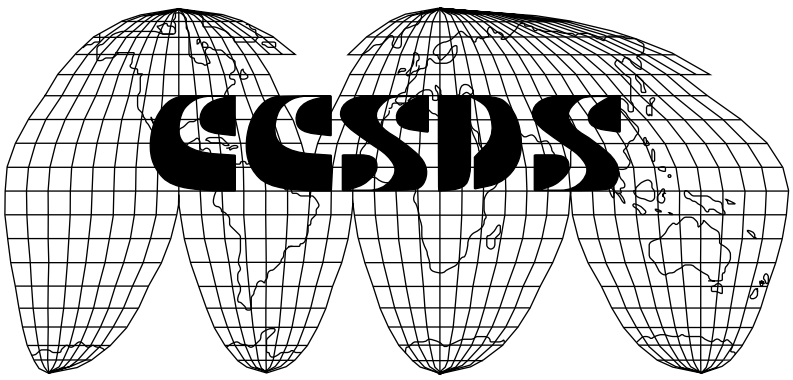
DRAFT RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

CCSDS FILE DELIVERY PROTOCOL (CFDP)

CCSDS 727.0-R-5

RED BOOK

August 2001



AUTHORITY

Issue:	Red Book, Issue 5
Date:	August 2001
Location:	Not Applicable

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS Recommendations is detailed in the reference [A1], and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code MT)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

- Whenever an Agency establishes a CCSDS-related standard, the Agency will provide other CCSDS member Agencies with the following information:

  -- The **standard** itself.

  -- The anticipated date of initial operational capability.

  -- The anticipated duration of operational service.

- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommendation** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or, (3) be retired or canceled.

In those instances when a new version of a **Recommendation** is issued, existing CCSDS-related Agency standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each Agency to determine when such standards or implementations are to be modified. Each Agency is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommendation.

FOREWORD

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING FOREWORD:)

Until relatively recently the typical storage medium for spacecraft has been the tape recorder, a complex device offering limited data storage and data access. The use of this type of storage has typically been limited to the recording and subsequent dump to the ground of telemetry data. Manipulation from the ground has required significant human intervention and used ad hoc, privately developed protocols.

The introduction of solid state mass memory providing gigabytes of storage with random access opens up a whole new ethos of spacecraft operation where much of the routine traffic to and from the spacecraft will be in the form of files. Furthermore, because of the random access nature of the onboard storage medium, it becomes possible to repeat transmission of data lost on the link and thus guarantee delivery of critical information.

To exploit the potential advantages of onboard mass memory, protocol support is required to provide a standard means to move data to and from the onboard storage medium in the form of files.

While the onboard storage medium has rapidly evolved, the essential constraints of space missions remain:

– limited systems resources in terms of computational power and memory capacities;

– environmental restrictions including noisy, bandwidth limited, asymmetrical, and interrupted communications links, some with very long propagation delay;

– varying user needs including a requirement for early access to transferred data regardless of its quality.

In view of these constraints, it is clear that there is a need for a file delivery service capable of transferring files to and from mass memory located in the space segment. Such a capability must not only operate under the constraints associated with space data communication, but it must also be applicable to the diverse range of mission configurations ranging from single low earth orbiting spacecraft to complex networks of relays, orbiters, and landers.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures which are defined in reference [A1]. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.
- Russian Space Agency (RSA)/Russian Federation.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Communications Research Centre (CRC)/Canada.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Federal Service of Scientific, Technical & Cultural Affairs (FSST&CA)/Belgium.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommendation. Its "Red Book" status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 727.0-R-1	CCSDS File Delivery Protocol, Red Book, Issue 1	June 1998	Original Issue (superseded)
CCSDS 727.0-R-2	CCSDS File Delivery Protocol, Red Book, Issue 2	March 1999	Superseded
CCSDS 727.0-R-3	CCSDS File Delivery Protocol, Red Book, Issue 3	July 1999	Stable Red Book issued during implementation and test phase; superseded.
CCSDS 727.0-R-4	CCSDS File Delivery Protocol, Red Book, Issue 4	February 2001	Superseded
CCSDS 727.0-R-5	CCSDS File Delivery Protocol, Red Book, Issue 5	August 2001	Current Issue: extended procedures have been removed from the main specification and placed in a non-normative annex (annex D).

CONTENTS

CONTENTS (continued)

Section Page

Figure

CONTENTS (continued)

Figure Page

Table

CONTENTS (continued)

Table Page

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

This document defines a CCSDS File Delivery Protocol (CFDP) and associated service for application in the space environment. It is intended for use over the current and envisaged packet delivery services used in the space environment including:

- CCSDS conventional packet telecommand;
- CCSDS conventional packet telemetry;
- CCSDS Advanced Orbiting Systems (AOS) Path service.

In may also operate over a wide variety of ground network services including those specified by the CCSDS for cross-support purposes.

The protocol operates in the space-to-ground, ground-to-space, and space-to-space directions of transfer. It may be initiated by the file sending or receiving entity.

In the interests of interoperability, protocol elements are included for generalized forms of standard file manipulation operations based on assumptions of a common model for a 'filestore', or medium used to store files. It is recognized, however, that the precise nature and capabilities of filestore management systems are operating-system dependent and, for that reason, the protocol assumes a virtual filestore and associated services that an implementation must map to the capabilities of the actual filestore used.

1.2 APPLICABILITY

This Recommendation specifies a protocol and associated services that are applicable to space missions with continuous duplex contact, intermittent duplex contact, asymmetrical time-disjunct contact, and simplex contact.

1.3 CONVENTIONS AND DEFINITIONS

1.3.1 BIT NUMBERING CONVENTION AND NOMENCLATURE

In this document, the following convention is used to identify each bit in an *N*-bit field. The first bit in the field to be transmitted (i.e., the most left justified when drawing a figure) is defined to be 'Bit 0'; the following bit is defined to be 'Bit 1' and so on up to 'Bit *N*–1'. When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, i.e., 'Bit 0'.

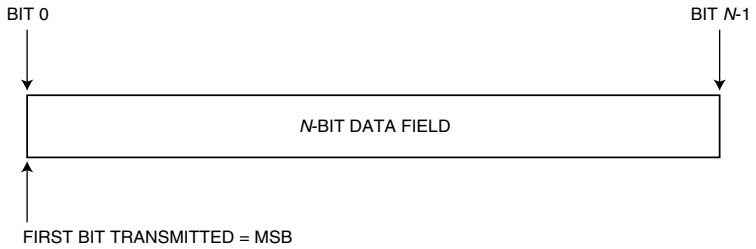


Figure 1-1: Bit Numbering Convention

In accordance with modern data communications practice, spacecraft data fields are often grouped into 8-bit 'words' which conform to the above convention. Throughout this Recommendation, the following nomenclature is used to describe this grouping:

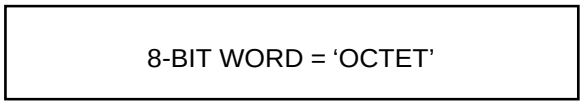


Figure 1-2: Octet Convention

By CCSDS convention, all 'spare' bits shall be permanently set to value 'zero'.

1.3.2 ORGANIZATION OF THE RECOMMENDATION

This Recommendation is organized as follows:

Section 2 provides an overview of the protocol, its intended use, and a description of the main interactions involved in a file transfer.

Section 3 defines the services provided by the protocol along with the associated primitives and parameters.

Section 4 is the main body of the protocol specification.

Section 5 defines the formats for the PDUs.

Section 6 defines proxy and directory operations.

Section 7 defines a number of classes of operation designed to meet the most typical operating scenarios.

Section 8 contains the state diagrams and state tables.

Section 9 describes the Management Information Base (MIB).

Annex A provides a list of informative references.

Annex B provides a list of acronyms and definitions.

Annex C provides an example of file checksum calculation.

Annex D documents proposed 'extended' procedures for conducting successive CFDP file copy operations among three or more entities in a store-and-forward manner.

1.3.3 DEFINITIONS

1.3.3.1 Definitions from OSI Basic Reference Model

This Recommendation makes use of a number of terms defined in reference [5]. The use of those terms in this Recommendation shall be understood in a generic sense, i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

- entity;
- Protocol Data Unit (PDU);
- service;
- Service Access Point (SAP);
- Service Data Unit (SDU).

1.3.3.2 Definitions from OSI Service Definition Conventions

This Recommendation makes use of a number of terms defined in reference [6]. The use of those terms in this Recommendation shall be understood in a generic sense, i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

- Indication;
- Primitive;
- Request;
- Response.

1.3.3.3 Terms Defined in This Recommendation

Within the context of this document the following definitions apply:

A *CFDP protocol entity* (or *CFDP entity*) is a functioning instance of an implementation of the CFDP protocol, roughly analogous to an Internet protocol 'host'.

The functional concatenation of a file and related *metadata* transmitted between two CFDP entities is termed a *File Delivery Unit* (FDU); in this context the term 'metadata' is used to

refer to any data exchanged between CFDP protocol entities in addition to file content, typically either additional application data (such as a 'message to user') or data that aid the recipient entity in effectively utilizing the file (such as file name). Note that an FDU may consist of metadata only. Note also that the term 'file' is frequently used in this specification as an abbreviation for 'file delivery unit'; only when the context clearly indicates that only actual files are being discussed, for example, in the explanation of the segmentation control parameter or the source and destination file name parameters of the CFDP Service Definition, should the term 'file' not be read as 'file delivery unit'.

The individual, bounded, self-identifying items of CFDP data transmitted between CFDP entities are termed *CFDP Protocol Data Units* or *CFDP PDU*s. Unless otherwise noted, in this document the term 'PDU' always means 'CFDP PDU'. CFDP PDUs are of two general types: *File Data PDUs* convey the contents of the files being delivered, while *File Directive PDUs* convey only metadata and other non-file information that advances the operation of the protocol.

A *transaction* is the end-to-end transmission of a single FDU between two CFDP entities. A single transaction normally entails the transmission and reception of multiple PDUs. Each transaction is identified by a unique transaction ID; all elements of any single FDU, both file content and metadata, are tagged with the same CFDP transaction ID.

Any single end-to-end file transmission task has two associated entities: the entity that has the file at the beginning of the task (the *source*) and the entity that has a copy of the file when the task is completed (the *destination*).

Each end-to-end file transmission task comprises a single point-to-point file copy operation. A file copy operation has two associated entities: the entity that has a copy of the file at the beginning of the operation (the *sender*) and the entity that has a copy of the file when the operation is completed (the *receiver*). (Note that file copy operations are functionally equivalent to file transmission tasks. The distinction between the two is preserved in this document solely as a terminological aid for possible future extension of the protocol to multi-point cases, in which a single file transmission task might comprise multiple successive file copy operations.)

Filestore is a generic term referring to the medium used to store files.

The term *offset* is used in a familiar way: the offset of a given octet of file data is the number of data octets that precede this octet in the file.

The file delivery *progress* represented by a given file data PDU is the sum of the offset of the PDU's file data content (the offset of the content's first octet) and the length of that file data content.

The *transmission progress* for a given transaction, delivering some file, is the maximum progress value over all file data PDUs sent so far in the course of this transaction.

Similarly the *reception progress* for a given transaction is the maximum progress value over all file data PDUs received so far in the course of this transaction. If and only if no data are lost in transmission, reception progress is equal to the number of file data octets received.

The term *CFDP user* is used to refer to the software task that causes the local entity to initiate a transaction or the software task that is notified by the local entity of the progress or completion of a transaction. The CFDP user local to the source entity is referred to as the *source CFDP user*. The CFDP user local to the destination entity is referred to as the *destination CFDP user*. The CFDP user may be operated by a human or by another software process. Unless otherwise noted the term *user* always refers to the CFDP user.

1.4 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommendations.

[1] *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*. Recommendation for Space Data System Standards, CCSDS 701.0-B-3. Blue Book. Issue 3. Washington, D.C.: CCSDS, June 2001.

[2] *Packet Telemetry*. Recommendation for Space Data System Standards, CCSDS 102.0-B-5. Blue Book. Issue 5. Washington, D.C.: CCSDS, November 2000.

[3] *Packet Telemetry Service Specification*. Recommendation for Space Data System Standards, CCSDS 103.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, June 2001.

[4] *Telecommand Part 2—Data Routing Service*. Recommendation for Space Data System Standards, CCSDS 202.0-B-3. Blue Book. Issue 3. Washington, D.C.: CCSDS, June 2001.

[5] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[6] *Information Technology—Open Systems Interconnection—Basic Reference Model—Conventions for the Definition of OSI Services*. International Standard, ISO/IEC 10731:1994. Geneva: ISO, 1994.

The latest issues of CCSDS documents may be obtained from the CCSDS Secretariat at the address indicated on page i.

2 OVERVIEW

2.1 GENERAL

This Recommendation defines a protocol suitable for the transmission of files to and from spacecraft data storage. In addition to the purely file delivery related functions, the protocol also includes file management services to allow control over the storage medium.

The protocol is capable of operating in a wide variety of mission configurations, from relatively simple low earth orbit spacecraft to complex arrangements of orbiters and landers supported by multiple ground facilities and transmission links.

The protocol is independent of the technology used to implement data storage and requires only a few fundamental filestore capabilities in order to operate. It assumes two filestores, one within the spacecraft and one on the ground, and operates by copying data between the two filestore locations.

The protocol makes no assumptions about the information being transferred and can be utilized for a wide range of applications involving the loading, dumping, and control of spacecraft storage.

The protocol has been specifically designed to minimize the resources required for operation. It is also scaleable, so that only those elements required to fulfill the selected options are required to be implemented.

Although the protocol can operate over a wide range of underlying communication services, this Recommendation assumes the use of CCSDS packet services as defined in references [1] through [4].

2.2 ARCHITECTURE ELEMENTS

2.2.1 GENERAL

The architectural elements involved in the file delivery protocol are depicted in figure 2-1 and described below.

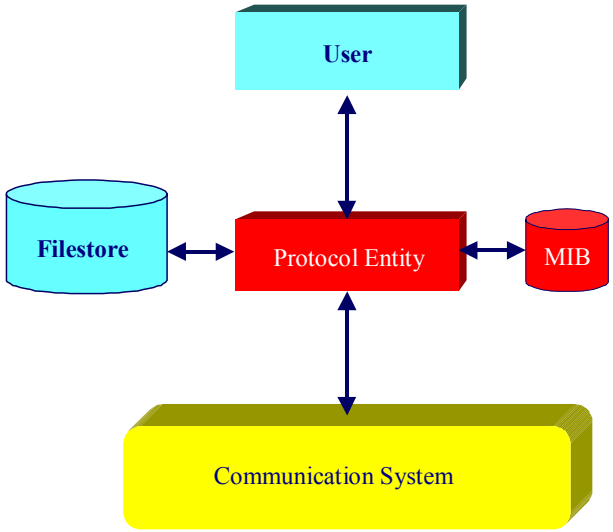


Figure 2-1: Architectural Elements of the File Delivery Protocol

2.2.2 USER

The protocol operates at the request of the CFDP user. The user interacts with the protocol using the service primitives defined in section 3.

A CFDP user is always a software task, which may or may not be operated by a human.

Each CFDP protocol entity has one (1) user.

2.2.3 PROTOCOL ENTITY

The protocol entity consists of implementations of the *delivery procedures*, which allow immediate file delivery and manipulation over a single network hop. A single service interface is presented to the user.

2.2.4 FILESTORE

The protocol operates by copying files from storage medium to storage medium, and it is therefore assumed that all CFDP entities have access to a local storage capability. As the ways in which the storage capability is provided will vary, the protocol is built on the premise that any file or organized set of files (i.e., a filestore) can be described in terms of a single standard representation. This representation, called a '**virtual filestore**' is assigned a

standard set of attributes which are then used by the protocol to manage the file delivery process. In an implementation, the virtual filestore must be mapped to and from actual hardware and software which constitute the real filestore. This approach allows complete independence from the technology used to implement the filestore.

2.2.5 UNDERLYING COMMUNICATION SYSTEM

The protocol assumes the availability of a single conceptual underlying communication system, referred to as the '**Unitdata Transfer (UT) layer**', to which all CFDP entities in a given CFDP addressing domain have access. In order that the protocol may operate over a wide range of implementations, the services required of the UT layer have intentionally been kept as simple as possible; those services are assumed to be made available to any single CFDP entity at only a single conceptual service access point.

Because of the potential diversity of physical underlying services in use, no reuse of underlying protocol features is utilized beyond the addressing required to identify localities within the domain of the underlying communication system ('**UT addresses**', to which CFDP entities' names may be mapped using information contained in the management information base) and the delimitation of the CFDP Protocol Data Unit (CFDP PDU).

The relative independence of the CFDP protocol specifically implies that all multiplexing to support multiple file delivery transactions occurs internally in the CFDP protocol entity and that sequence auditing and error detection is provided by CFDP. PDU length information is incorporated in the CFDP PDU header to give independence from the underlying protocol and to simplify protocol processing.

It should be noted that although the CFDP protocol can operate over a service where data errors, data loss, and out-of-sequence delivery occur, it is not intended to compensate for networks where these effects are prevalent. Severe performance reductions will result if such an approach is taken. A simplified analysis of CFDP performance is detailed in the CFDP Green Book (CCSDS 720.1-G-1). This analysis will allow implementers to gauge likely performance based on file size, file segment size and bit error probability.

2.2.6 MANAGEMENT INFORMATION BASE

To perform a file delivery, a significant amount of information must be passed by the local user to its local CFDP entity and by the local CFDP entity to the remote CFDP entity. Typically, this data is static and may be stored within the CFDP entities as system tables, referred to as the *Management Information Base* (*MIB*). The MIB contains such information as default values for user communications requirements, e.g., for address mapping, and for communication timer settings. The MIB is formally defined as part of the protocol specification in section 9.

2.3 GENERAL CHARACTERISTICS

2.3.1 GENERAL

The CFDP enables the moving of a file from one filestore to another, where the two filestores are in general resident in separate data systems and often with an intervening space link. Functionally attached to each such file may be metadata of variable size, format, and semantics. The metadata consists of information associated with the file or the file transfer such as file names, filestore requests, or messages to the CFDP user. The functional concatenation of a file and all relevant metadata is termed an FDU, but it is also permissible for an FDU to consist of metadata only.

FDU transmission is accomplished by CFDP entities, each of which has access to exactly one filestore. A transaction is the end-to-end transmission of a single FDU between two CFDP entities. The initiation of the transaction causes a file copy operation to occur between two entities, the sender and the receiver.

As depicted in figure 2-2, the protocol procedures constitute the interaction between two protocol entities. The sending entity is the entity from which the file is copied in a file copy operation. The receiving entity is the entity to which the file is copied in a file copy operation.

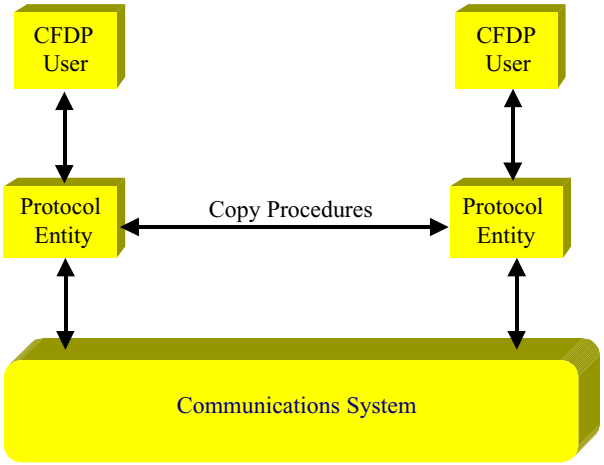


Figure 2-2: CFDP Procedures

The reliability of a transaction is determined by whether the transaction is chosen to operate in unacknowledged mode or one of the acknowledged modes. In unacknowledged mode, data delivery failures are not reported to the sender and therefore cannot be repaired. Reception of the complete file is therefore not guaranteed. In acknowledged mode, the receiver informs the sender of any undelivered file segments or ancillary data. These are then retransmitted, guaranteeing complete file delivery.

Each transaction results in the copying of a single FDU from source to destination. Within any single transaction, the subsidiary actions cancel, suspend, resume, and report may be performed.

The transaction is terminated when one of the following conditions applies:

- the file has been successfully transferred;
- a sending or receiving entity has disallowed the transaction;
- the transaction has been canceled because a fault was detected;
- the source or destination CFDP user has canceled the transaction.

2.3.2 USER OPERATIONS

The term 'user operations' refers to the use of the CFDP services offered by the local CFDP entity to cause the CFDP user of a remote CFDP entity to initiate additional CFDP transactions. User operations are implemented using the 'Message to User' capability of the protocol to forward an 'order' to the remote CFDP user which will in turn initiate a transaction with its local CFDP entity.

Five standard user operations are defined: proxy operations, remote status report operations, remote suspend and resume operations, and directory operations. Directory operations are used to request a listing of the contents of a specified directory in the remote user's local filestore. Remote status report operations are used to request a report of the status of a specified CFDP transaction at the remote entity. Remote suspend and resume operations are used to request the suspension and resumption of a specified CFDP transaction at the remote entity. Proxy operations are used to initiate the delivery of a file from a remote CFDP entity to some other user, either to the local user itself (in which case the proxy operation functions as a 'Get') or to the user of some third CFDP entity. The FDU transmitted in a proxy operation normally contains a file but may contain only metadata, such as filestore directives or a Message to User containing an order to another remote CFDP user.

User operations are described in section 6.

2.3.3 ADDRESSING

Within any single CFDP network (i.e., any set of CFDP entities that is closed under communication), each CFDP entity is assigned a unique name. As each CFDP entity has access to exactly one filestore, identification of a CFDP entity implicitly identifies the associated filestore. At each CFDP entity location, address look-up capabilities are provided using information contained in the associated MIB. This look-up capability provides translation between the network-unique name of a CFDP entity and the corresponding UT address, which may in reality be an Internet address, radio device buffer, APID, virtual channel number, or other implementation-specific mechanism.

2.3.4 RETRANSMISSION STRATEGIES

The quality of service offered by the protocol is selectable, according to mission requirements and transmission capability, and ranges from an unacknowledged option, whereby a file is transmitted with no attempt at completeness should errors occur (errors will be detected and data discarded), to a fully acknowledged option providing error recovery through retransmission. For the acknowledged modes of operation, several sub-options may be selected by the receiver. These sub-options relate to release time of any Negative Acknowledgments (NAKs) and range from immediate release to deferred release (whereby any NAKs are stored until the assumed end of the transmission). The extent of available retransmission strategies is more completely described in section 4.

2.3.5 VIRTUAL FILESTORE

The virtual filestore concept provides for a mapping of protocol filestore directives to actual filestore manipulation. The way in which this mapping is performed in an implementation is a local matter. Allowances are made in the protocol for inclusion of additional filestore directives using the Type-Length-Value (TLV) capability of the Metadata PDU.

To enable interoperability, this Recommendation assumes a minimum set of capabilities from the virtual filestore as follows:

- create file;
- delete file;
- rename file;
- append file;
- replace file;
- create directory;
- remove directory;
- list directory.

In some circumstances, it is advantageous for the CFDP protocol to be able to recognize record boundaries within the file. If this option is to be used, the filestore must have the capability to make the distinction between such files and those which are to be treated as a stream of octets.

2.4 OVERVIEW OF INTERACTIONS

Figure 2-3 shows the details of a Copy operation. (In all following figures in this section, dashed lines denote optional actions.)

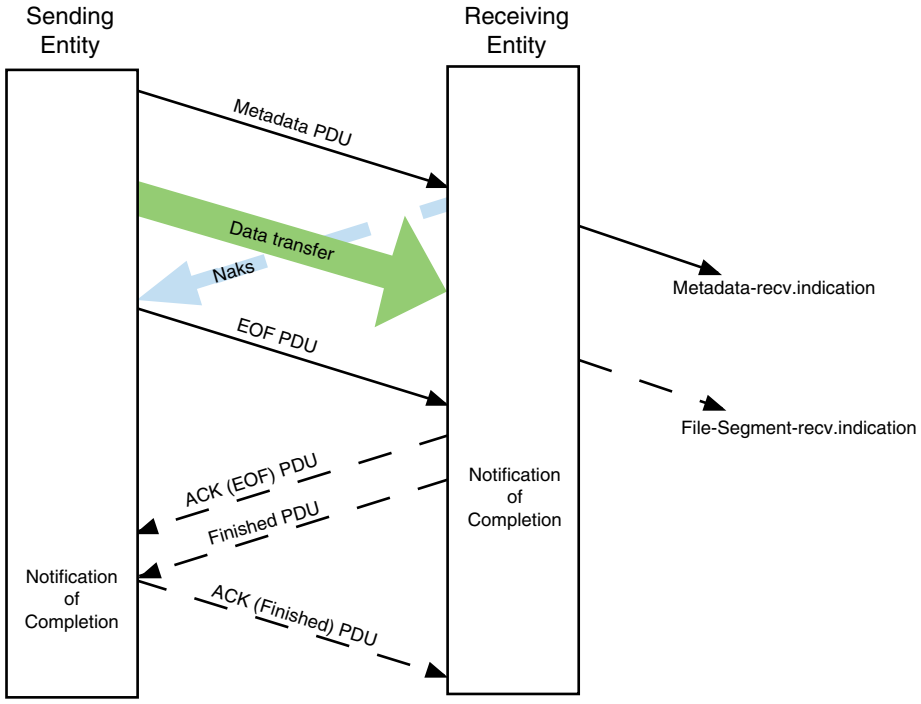


Figure 2-3: Copy Operations, Sequence of Events

All copy operations start with transmission of the Metadata, followed by transmission of the file segments and a PDU to indicate the End of File (EOF). Optional service primitives may be issued at the receiving entity to announce receipt of individual file segments; notification of receipt of Metadata is mandatory.

For reliable transactions, a number of features are added. NAKs are used to request retransmission of lost data. Receipt of the EOF PDU is ensured via an EOF acknowledgement. Since lost data may still be outstanding after the EOF sequence, a Finished PDU is sent when all file data has been successfully assembled. Delivery of this PDU is ensured through use of the Finished acknowledgement.

When the file transmission task is deemed complete the sending and receiving entities issue Notices of Completion. These Notices of Completion are abstract internal events which are not exposed to the CFDP user. They are used to inform the Put procedures (described later in this section) of completion of the file transmission task.

The Put transaction is initiated by a `Put.request` from a CFDP user to the source entity. The `Put.request` results in initiation of the FDU transmission procedure and also of the first (possibly only) copy operation. The `Put.request` also results in the notification of a

Transaction identifier to the application so that the transaction may subsequently be unambiguously identified. A Notice of Completion at the source entity causes it to issue a `Transaction-Finished.indication`. This process is optional at the destination entity. The Put transaction is illustrated in figure 2-4.

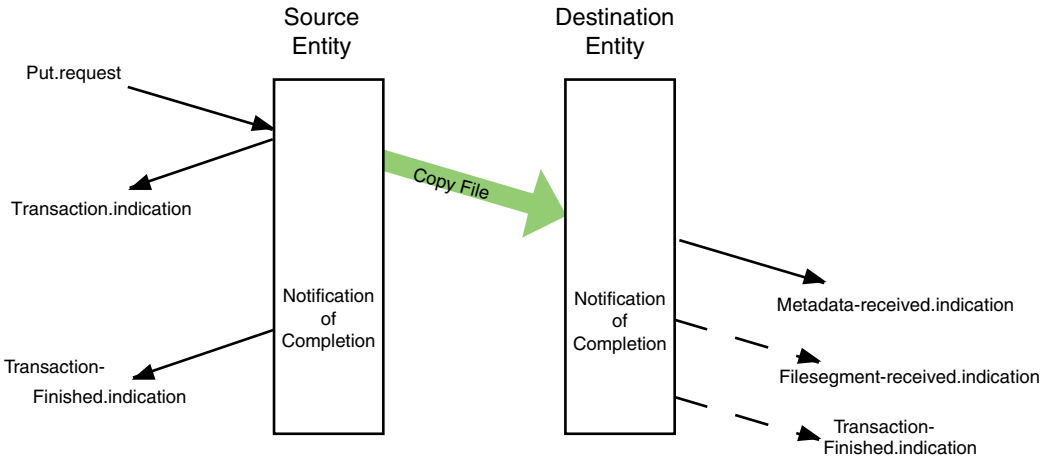


Figure 2-4: Put Operations, Sequence of Events

When the source of the file to be transferred is other than the local filestore, the requesting user must initiate a Put transaction containing a message to the user of the remote CFDP entity, the 'proxy user'. This message requests that the proxy user in turn submit a `Put.request` to that entity. The destination of the requested Put transaction may be the original requesting user (in which case the proxy operation functions as a 'Get') or some third CFDP user. On completion of the second transaction, the proxy user initiates a third transaction to notify the original requesting user of the completion of the proxy operation. A proxy put operation is illustrated in figure 2-5.

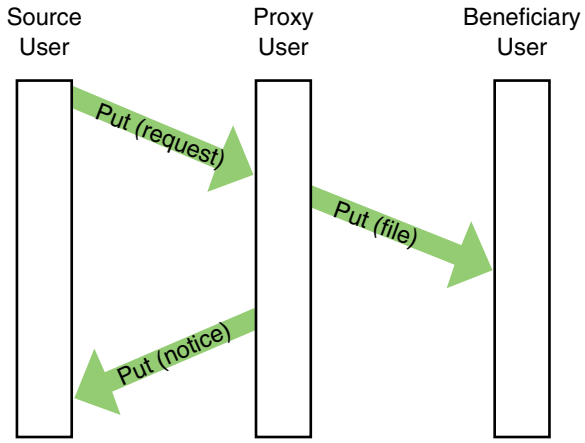


Figure 2-5: Proxy Put Operations, Sequence of Events

3 SERVICE DESCRIPTION

3.1 SERVICES AT THE USER INTERFACE

3.1.1 The services provided by the protocol shall be made available to the CFDP user and shall include the following:

a) initiating the transmission of files between filestores;

b) initiating filestore operations on remote filestores;

c) receiving events related to the operation of current transactions;

d) requesting status information related to the current transactions;

e) sending or receiving messages associated with the current transactions;

f) suspending, resuming, or canceling the transmission of a current transaction.

3.1.2 The CFDP entity shall be implemented such that virtually any number of transactions may be concurrently in various stages of transmission or reception at a single CFDP entity.

NOTE – To clarify: the implementation must be able to accept a `Put.request` primitive, and thereupon initiate a new transaction, prior to the completion of previously initiated transactions. The completion of a transaction encompasses the initial transmission of file data by the source entity, the reception of some or all of that data by the destination entity, and any consequent acknowledgment and retransmission data exchange that may be necessary between the two entities. The requirement for concurrent transaction support therefore does not necessarily imply that the implementation must be able to begin initial transmission of file data for one transaction while initial transmission of file data for one or more other transactions is still in progress. (But neither is support for this functional model precluded.)

3.2 SUMMARY OF PRIMITIVES

3.2.1 The CFDP service shall consume the following request primitives:

a) `Put.request`;

b) `Report.request`;

c) `Cancel.request`;

d) `Suspend.request`;

e) `Resume.request`.

3.2.2 The CFDP service shall deliver the following indication primitives:

a) `Transaction.indication`;

b) `EOF-Sent.indication`;

c) `Transaction-Finished.indication`;

d) `Metadata-Recv.indication`;

e) `File-Segment-Recv.indication`;

f) `Report.indication`;

g) `Suspended.indication`;

h) `Resumed.indication`;

i) `Fault.indication`;

j) `Abandoned.indication`.

3.3 SUMMARY OF PARAMETERS

NOTE – The availability and use of parameters for each primitive are indicated in 3.5, where parameters that are optional are identified with square brackets [thus]. The following definitions apply.

3.3.1 The *destination CFDP entity ID* parameter shall uniquely identify the CFDP entity **to** which the FDU is to be sent.

3.3.2 The *source CFDP entity ID* parameter shall uniquely identify the CFDP entity **from** which the FDU is to be sent.

3.3.3 The *source file name* parameter

a) shall contain the full path name at which the file to be copied is located at the filestore associated with the source entity;

b) shall be omitted when the FDU to be Put contains only metadata, such as a message to a user or a standalone filestore request.

3.3.4 The *destination file name* parameter

a) shall contain the full path name to which the file to be copied will be placed at the filestore associated with the destination entity;

b) shall be omitted when the FDU to be Put contains only metadata, such as a message to a user or a standalone filestore request.

3.3.5 The *segmentation control* parameter

a) shall indicate whether the file being delivered is to be segmented as an array of octets or as an array of variable-length records;

b) shall be omitted when local and remote file names are omitted.

3.3.6 The *Transaction ID* parameter shall uniquely identify a single instance of FDU delivery and shall contain the ID of the source CFDP entity together with a sequence number that is specific to that entity.

NOTES

1 At any moment, any given transaction ID is unique within the CFDP addressing domain that encompasses the source CFDP entity.

2 The entity ID and transaction sequence number are variable in length to accommodate differing mission requirements. A common parsing scheme is defined to allow full interoperability despite differing parameter lengths.

3.3.7 The *Condition code* parameter shall provide additional information on some change in transaction status.

3.3.8 The *Status report* parameter shall indicate the status of the indicated file delivery transaction. The format and scope of the *status report* parameter are specific to the implementation. It could contain information such as:

a) whether the transaction is finished, canceled, suspended, or active;

b) what extents of the FDU are known to have been successfully received by the receiving CFDP entity;

c) what extents of the FDU are known to have been transmitted by the sending CFDP entity.

3.3.9 *Fault handler overrides*:

a) If included, the optional *fault handler overrides* shall indicate the actions to be taken upon detection of one or more types of fault condition. Each fault handler override shall identify both a type of fault condition to be handled and the action to be taken in the event that a fault of this type is detected; each such action shall be one of the following:

   – cancel the file delivery transaction,

   – suspend the transaction,

   – ignore the fault but issue a `Fault.indication` primitive to the local CFDP user, or

   – abandon the transaction and issue an `Abandoned.indication` primitive to the local CFDP user.

b) The fault conditions for which handler overrides may be specified are a subset of the conditions listed in table 5-5, later in this specification. The listed conditions that are *not* fault conditions are 'No error', 'Suspend request received', and 'Cancel request received'.

c) For each type of fault condition for which no *fault handler override* is included among the service request parameters, a default handler shall be selected according to the contents of the MIB.

3.3.10 If included, the optional *transmission mode* parameter shall override the default transmission mode. The values of the transmission mode parameter are 'acknowledged' or 'unacknowledged'.

3.3.11 If included, the optional *Messages to User* parameter shall be transmitted at the beginning of the transaction and delivered to the destination CFDP user upon receipt. Certain messages are defined in the Proxy, Remote Status Report, and Directory Operations section to allow remote initiation of CFDP transactions.

3.3.12 *Filestore Requests*:

a) if included, the optional *filestore requests* shall be transmitted at the beginning of the transaction and shall be acted upon by the destination entity when all data transfer activities of the transaction are completed;

b) to indicate the action to be taken, each *filestore request* shall include sub-parameters as follows:

- *action*,
- *file name 1*,
- *file name 2* (present for certain values of the *action* parameter);

c) the *action* sub-parameter may take the values of:

- *create file*,
- *delete file*,
- *rename file (file name 2* present*)*,
- *append file* (*file name 2* present),
- *replace file* (*file name 2* present),
- *create directory*,
- *remove directory*,
- *'deny' file* (like *delete*, but does not fail if file does not exist);

d) when the value of the *file name 1* or *file name 2* sub-parameter is equal to the value of the *destination file name* parameter, the file to which the filestore request pertains shall be the newly delivered file;

e) in the case of multiple *filestore requests* in the same transaction, the source entity shall preserve the order in which the requests were submitted and the destination entity shall execute them in this same order.

3.3.13 The *Filestore Responses* parameter shall indicate whether the indicated *filestore requests* in the indicated transaction succeeded or failed and shall include the following additional information for each filestore request:

– if it failed, an explanation;

– if it succeeded, a report in action-specific format (if relevant).

3.3.14 The *Filestore Responses* parameter shall not be used to convey directory listing information; the transfer of directory listing information is achieved via Directory Operations transactions.

3.3.15 The *flow label* parameter may optionally be used to support prioritization and preemption schemes.

NOTE – The *flow label* parameter should be taken as a hint to the order in which PDUs should be transmitted when the opportunity arises, but the manner in which flow labels are used is strictly an implementation matter.

3.3.16 The *offset* parameter shall indicate a displacement from the beginning of the file.

3.3.17 The *length* parameter shall indicate the number of octets of file data received.

3.3.18 The *progress* parameter shall report on current file transmission or reception progress, as defined earlier.

3.4 SERVICES REQUIRED OF THE UNDERLYING COMMUNICATION SYSTEM

The service primitives and parameters required to access the services of the Unitdata Transfer (UT) layer are:

`UNITDATA.request`	(UT_SDU, UT Address)
`UNITDATA.indication`	(UT_SDU, UT Address)

NOTES

1 The service assumed of the underlying layer is as simple as possible to allow CFDP to be as generally applicable as possible. For the purposes of this specification the service is referred to as the UT service and the delimited data unit it transfers is the UT Service Data Unit (UT_SDU). Although the UT layer is conceptually a single service, in practice CFDP might use several different physical underlying communication systems at different times, or even concurrently, depending on the remote entities with which it is in communication. In a purely CCSDS network the sole service could be the CCSDS Path or Packet service, and the UT_SDU would be the data field of a CCSDS packet.

2 The format and contents of the UT Address parameter depend on the addressing capabilities and conventions of the underlying service. Information in the MIB must enable translation between CFDP entity names and the corresponding UT addresses.

3 The CFDP protocol operates over a single conceptual UT service access point. In practice, the implementation of the UT layer is not constrained to provide all services through a single physical service access point. Different physical service access points would necessarily be provided if different physical underlying communication systems were in use concurrently. A given implementation might even maintain a different physical service access point for each remote entity with which CFDP is currently (at any moment) in direct communication.

4 The above list of mandatory UT layer service primitives does not imply that the UT layer is prohibited from supporting additional services for the convenience of the CFDP implementation. Possible additional primitives that might prove valuable include:

- An indication whenever the UT layer is prepared to accept another PDU for transmission to the UT layer instance at an identified UT address. (This indication could be used to implement data flow control. Destination UT address might be needed in the indication because the UT layer might be transmitting concurrently to multiple UT addresses.)
- An indication whenever the UT layer completes transmission of a UT_SDU whose content is a CFDP PDU to which Positive Acknowledgment procedures apply. (This indication could be used to start positive acknowledgment timers.)

5 The assumed minimum underlying quality of service is:

- with possible errors in the delivered UT_SDUs;
- incomplete, with some UT_SDUs missing;
- in sequence; i.e., the delivered UT_SDUs are delivered in the order in which they were transmitted. (In-sequence arrival is specifically assumed in the design of the immediate, asynchronous, and prompted lost segment detection modes and the Keep Alive mechanism.)

The CFDP protocol can operate over a service where data errors and data loss occur; and when no incremental lost segment detection procedures are in effect, even out-of-sequence data arrival can be tolerated. However, the protocol is not intended to compensate for networks where these effects are prevalent. Severe performance reductions will result if such an approach is taken.

3.5 CFDP SERVICE PRIMITIVES

3.5.1 Put.request

3.5.1.1 Function

The Put.request primitive shall be used by the application to request delivery of a file from the source filestore to a destination filestore.

3.5.1.2 Semantics

Put.request shall provide parameters as follows:

Put.request	(destination CFDP entity ID, [source file name], [destination file name], [segmentation control], [fault handler overrides], [flow label], [transmission mode], [messages to user], [filestore requests])

3.5.1.3 When Generated

Put.request is generated by the source CFDP user at any time.

3.5.1.4 Effect on Receipt

Receipt of Put.request shall cause the CFDP entity to initiate source entity put procedures.

3.5.1.5 Additional Comments

None.

3.5.2 Cancel.request

3.5.2.1 Function

The Cancel.request primitive shall be used to request that a transaction be canceled.

3.5.2.2 Semantics

Cancel.request shall provide parameters as follows:

Cancel.request (transaction ID)

3.5.2.3 When Generated

Cancel.request is generated by any CFDP user at any time during the lifetime of any transaction.

3.5.2.4 Effect on Receipt

Receipt of Cancel.request shall cause a Notice of Cancellation at the local entity.

3.5.2.5 Additional Comments

None.

3.5.3 Suspend.request

3.5.3.1 Function

The Suspend.request primitive shall be used to request that a transaction be suspended.

3.5.3.2 Semantics

Suspend.request shall provide parameters as follows:

Suspend.request (transaction ID)

3.5.3.3 When Generated

Suspend.request is generated by any CFDP user at any time during the lifetime of any transaction, except that for a transaction sent in unacknowledged mode it can only be generated by the transaction's source user.

3.5.3.4 Effect on Receipt

Receipt of Suspend.request shall cause a Notice of Suspension at the local entity.

3.5.3.5 Additional Comments

None.

3.5.4 Resume.request

3.5.4.1 Function

The Resume.request primitive shall be used to request that a suspended transaction be resumed.

3.5.4.2 Semantics

Resume.request shall provide parameters as follows:

Resume.request (transaction ID)

3.5.4.3 When Generated

Resume.request is generated by any CFDP user at any time during the lifetime of any transaction, except that for a transaction sent in unacknowledged mode it can only be generated by the transaction's source user.

3.5.4.4 Effect on Receipt

Receipt of Resume.request shall have no effect on any transaction that is not currently suspended. For a transaction that is currently suspended, it shall cause the local CFDP entity to initiate Resume procedures.

3.5.4.5 Additional Comments

User application developers may find it advisable to provide automated support for one special case of transaction resumption, which applies to any transaction which the user application can determine is currently suspended solely due to transaction inactivity (i.e., as a result of an Inactivity fault). On receipt of any service indication produced because of arrival of a PDU for such a transaction, the user application might infer that the transaction's inactivity is ended and thereupon automatically resume the transaction.

3.5.5 Report.request

3.5.5.1 Function

The Report.request primitive shall be used to request a report on the status of a transaction.

3.5.5.2 Semantics

Report.request shall provide parameters as follows:

Report.request (transaction ID)

3.5.5.3 When Generated

Report.request is generated by any CFDP user at any time during the lifetime of any transaction.

3.5.5.4 Effect on Receipt

Receipt of Report.request shall cause the CFDP entity to return a Report.indication primitive to the CFDP user.

3.5.5.5 Additional Comments

None.

3.5.6 Transaction.indication

3.5.6.1 Function

The Transaction.indication primitive shall be used to indicate the Transaction identifier to the source CFDP user.

3.5.6.2 Semantics

Transaction.indication shall provide parameters as follows:

Transaction.indication (transaction ID)

3.5.6.3 When Generated

Transaction.indication shall be generated on receipt of a Put.request primitive.

3.5.6.4 Effect on Receipt

The effect on receipt of Transaction.indication by a CFDP user is undefined.

3.5.6.5 Additional Comments

The Transaction ID parameter is returned by the CFDP entity and provides the source CFDP user with a means of uniquely identifying the associated transaction thereafter.

3.5.7 EOF-Sent.indication

3.5.7.1 Function

The EOF-Sent.indication primitive shall be used to notify the source CFDP user of the initial transmission of a transaction's EOF PDU.

3.5.7.2 Semantics

EOF-Sent.indication shall provide parameters as follows:

EOF-Sent.indication (transaction ID)

3.5.7.3 When Generated

EOF-Sent.indication shall be generated on initial transmission of an EOF PDU, at the conclusion of initial transmission of a transaction's Metadata and file data.

3.5.7.4 Effect on Receipt

The effect on receipt of EOF-Sent.indication by a CFDP user is undefined.

3.5.7.5 Additional Comments

Generation of EOF-Sent.indication is optional. The user may choose to base a flow control system in part on this mechanism, e.g., to refrain from submitting a new Put.request primitive until the EOF-Sent.indication resulting from the previous one has been received.

3.5.8 Transaction-Finished.indication

3.5.8.1 Function

The Transaction-Finished.indication primitive shall be used to indicate to the source or destination CFDP user that the transaction is complete and that the source CFDP user is authorized to modify or delete any retransmission buffer (file) that it was sharing with the protocol entity to conserve persistent storage space.

3.5.8.2 Semantics

Transaction-Finished.indication shall provide parameters as follows:

Transaction-Finished.indication	(transaction ID, [filestore responses], [status report], condition code)

3.5.8.3 When Generated

Transaction-Finished.indication is generated on Notice of Completion of a file transmission procedure either at the source CFDP entity or, optionally, at the destination CFDP entity.

3.5.8.4 Effect on Receipt

The effect on receipt of Transaction-Finished.indication by an application is undefined.

3.5.8.5 Additional Comments

Transaction-Finished.indication is always generated by the source CFDP entity and can also be generated by the destination CFDP entity. The condition code indicates the condition under which the transaction was finished.

3.5.9 Metadata-Recv.indication

3.5.9.1 Function

The Metadata-Recv.indication primitive shall be used to indicate to the destination CFDP user that the metadata associated with a transaction has been received.

3.5.9.2 Semantics

Metadata-Recv.indication shall provide parameters as follows:

Metadata-Recv.indication	(transaction ID, source CFDP entity ID, [source file name], [destination file name], [messages to user])

3.5.9.3 When Generated

Metadata-Recv.indication is generated on receipt of a Metadata PDU.

3.5.9.4 Effect on Receipt

The effect on receipt of Metadata-Recv.indication by the destination CFDP user is undefined.

3.5.9.5 Additional Comments

None.

3.5.10 File-Segment-Recv.indication

3.5.10.1 Function

The File-Segment-Recv.indication primitive shall be used to indicate, to the destination CFDP user, the receipt of individual file data (non metadata) segments.

3.5.10.2 Semantics

File-Segment-Recv.indication shall provide parameters as follows:

File-Segment-Recv.indication	(transaction ID, offset, length)

3.5.10.3 When Generated

File-Segment-Recv.indication is generated on receipt of a File Data PDU.

3.5.10.4 Effect on Receipt

The effect on receipt of File-Segment-Recv.indication by a CFDP user is undefined.

3.5.10.5 Additional Comments

Generation of File-Segment-Recv.indication is optional. Offset is the number of octets in the file prior to the first octet of the received File Data PDU's content.

3.5.11 Suspended.indication

3.5.11.1 Function

The Suspended.indication primitive shall be used to indicate to the CFDP user that the transaction has been suspended.

3.5.11.2 Semantics

Suspended.indication shall provide parameters as follows:

Suspended.indication	(transaction ID, condition code)

3.5.11.3 When Generated

Suspended.indication is generated on Notice of Suspension of a file transmission procedure.

3.5.11.4 Effect on Receipt

The effect on receipt of Suspended.indication by a CFDP user is undefined.

3.5.11.5 Additional Comments

None.

3.5.12 Resumed.indication

3.5.12.1 Function

The Resumed.indication primitive shall be used to indicate to the CFDP user that the transaction has been resumed.

3.5.12.2 Semantics

Resumed.indication shall provide parameters as follows:

Resumed.indication	(transaction ID, progress)

3.5.12.3 When Generated

Resumed.indication shall be generated in response to Resume.request.

3.5.12.4 Effect on Receipt

The effect on receipt of Resumed.indication by a CFDP user is undefined.

3.5.12.5 Additional Comments

Progress indicates how far the issuing CFDP entity had progressed, in sending or receiving the transaction's transmitted file, as of the moment at which the Resumed.indication was generated.

3.5.13 Report.indication

3.5.13.1 Function

The Report.indication primitive shall be used to report to the CFDP user on the status of a transaction.

3.5.13.2 Semantics

Report.indication shall provide parameters as follows:

Report.indication	(transaction ID, status report)

3.5.13.3 When Generated

Report.indication shall be generated on receipt of the Report.request primitive.

3.5.13.4 Effect on Receipt

The effect on receipt of Report.indication by a CFDP user is undefined.

3.5.13.5 Additional Comments

The format and scope of the *status report* parameter are specific to the implementation.

3.5.14 `Fault.indication`

3.5.14.1 Function

The `Fault.indication` primitive shall be used to indicate to the CFDP user the occurrence of a fault condition for which the designated fault handler was 'Ignore'.

3.5.14.2 Semantics

`Fault.indication` shall provide parameters as follows:

`Fault.indication`	(transaction ID, condition code, progress)

3.5.14.3 When Generated

`Fault.indication` shall be generated upon detection of a fault condition for which the designated fault handler is 'Ignore'.

3.5.14.4 Effect on Receipt

The effect on receipt of `Fault.indication` by a CFDP user is undefined.

3.5.14.5 Additional Comments

The progress parameter indicates how far the issuing CFDP entity had progressed, in sending or receiving the transaction's transmitted file, as of the moment at which the `Fault.indication` was generated.

3.5.15 Abandoned.indication

3.5.15.1 Function

The Abandoned.indication primitive shall be used to indicate to the CFDP user the occurrence of a fault condition for which the designated fault handler was 'Abandon'.

3.5.15.2 Semantics

Abandoned.indication shall provide parameters as follows:

Abandoned.indication (transaction ID,
condition code,
progress)

3.5.15.3 When Generated

Abandoned.indication shall be generated upon detection of a fault condition for which the designated fault handler is 'Abandon'.

3.5.15.4 Effect on Receipt

The effect on receipt of Abandoned.indication by a CFDP user is undefined.

3.5.15.5 Additional Comments

The progress parameter indicates how far the issuing CFDP entity had progressed, in sending or receiving the transaction's transmitted file, as of the moment at which the Abandoned.indication was generated.

4 PROTOCOL SPECIFICATION

4.1 PROCEDURES

4.1.1 CRC PROCEDURES

4.1.1.1 CRC Procedures at the PDU Transmitting Entity

If the CRC option is active, the PDU transmitting entity shall set the CRC flag to 'true' and calculate and insert the CRC for each outgoing PDU.

4.1.1.2 CRC Procedures at the PDU Receiving Entity

If the CRC flag is set to 'true' in the incoming PDU, the PDU receiving entity shall calculate the CRC and discard the PDU if it does not match the CRC field in the PDU header.

4.1.1.3 CRC Algorithm

4.1.1.3.1 The CRC computation algorithm shall be the standard CCSDS Telecommand CRC algorithm as specified in 4.2.1.3 of the CCSDS Telecommand Recommendation (reference [4]).

4.1.1.3.2 The algorithm shall be applied as follows:

- the CRC field in the PDU header shall be set to 'zero';
- the CRC value for the entire PDU (including the zero CRC value) shall then be calculated;
- the calculated CRC value shall then be inserted into the CRC field in the PDU header.

4.1.2 PUT PROCEDURES

Receipt of a `Put.request` primitive shall cause the source CFDP entity to initiate

a) a Transaction Start Notification procedure; and

b) a Copy File procedure, for which

   1) any fault handler overrides shall be derived from the `Put.request`,

   2) any Messages to User or filestore requests shall be derived from the `Put.request`,

   3) omission of source and destination filenames shall indicate that only Metadata will be delivered,

   4) the transmission mode (acknowledged or unacknowledged) is defined by the existing content of the MIB unless overridden by the transmission mode parameter of the `Put.request`.

4.1.3 TRANSACTION START NOTIFICATION PROCEDURE

4.1.3.1 The transaction identifier shall be

a) issued sequentially by the source CFDP entity in response to a service request; and

b) used for all subsequent references to the transaction.

4.1.3.2 No two transactions for which there is any contemporaneous protocol activity may have the same transaction identifier.

NOTE – Duplicate transaction identifiers will inevitably result in erroneous protocol behavior.

4.1.3.3 The source CFDP entity shall issue a `Transaction.indication` primitive that notifies the source CFDP user of the transaction identifier.

4.1.4 PDU FORWARDING PROCEDURES

4.1.4.1 Each outgoing PDU shall be directed to the appropriate CFDP entity using the addressing capabilities of the underlying Unitdata Transfer service.

4.1.4.2 The remote Unitdata Transfer Service Access Point (UT-SAP) to which the PDU is to be directed shall be obtained from the MIB, based on the identifier of the CFDP entity to which the PDU is to be delivered.

4.1.4.3 Depending on the type of PDU, the identifier of the CFDP entity to which the PDU is to be delivered shall be one of the following:

– for an ACK PDU, the ID of the CFDP entity which sent the PDU that is being acknowledged;

– for a File Data, Metadata, EOF, or Prompt PDU, the receiver entity ID for that file copy operation;

– for a NAK, Finished, or Keep Alive PDU, the sender entity ID for that file copy operation.

4.1.5 COPY FILE PROCEDURES

NOTES

1 In the following subsections, the name of an EOF PDU or a Finished PDU includes, in parentheses, either the name of the specific condition code that the PDU carries (such as ‘No error’) or the general condition name ‘cancel’ indicating that the PDU carries some condition code other than ‘No error’ and ‘Suspend request received’.

2 The name of an ACK PDU includes, in parentheses, the type of PDU which is being acknowledged: EOF or Finished.

4.1.5.1 General Procedures

4.1.5.1.1 Copy File Procedures at the Sending Entity

4.1.5.1.1.1 Initiation of the Copy File procedures shall cause the sending CFDP entity to forward a Metadata PDU to the receiving CFDP entity.

4.1.5.1.1.2 The Metadata PDU shall contain

a) an indication of whether or not the file contains records whose boundaries are to be respected when the file is segmented for transmission in file data PDUs;

b) the size of the file if known (i.e., for a bounded file);

c) the source and destination names (path names) of the file;

d) optional fault handler overrides, Messages to User, filestore requests, and/or flow label.

NOTE – Assuring that all relevant metadata for the transaction are contained within a single Metadata PDU is an implementation responsibility.

4.1.5.1.1.3 For transactions that deliver more than just metadata, Copy File initiation also shall cause the sending CFDP entity to retrieve the file from the sending filestore and to transmit it in File Data PDUs.

4.1.5.1.1.4 Each segment of data shall be forwarded in a File Data PDU, which also shall contain the offset, in octets, from the beginning of the file at which the segment is located.

4.1.5.1.1.5 If the *segmentation control* service parameter has requested that record boundaries be respected, the first octet of a segment shall always be the first octet of a file record. Multiple complete records may occupy a single segment, but no record may span multiple segments.

NOTE – The method by which record boundaries in a file are detected is a local implementation matter. Agreement on such a method is not necessary for interoperability among CFDP entities. (However, it may be required for the correct operation of the source and destination CFDP user applications.)

4.1.5.1.1.6 If the *segmentation control* service parameter was omitted, record boundaries shall by default not be respected.

NOTE – The segmentation algorithm is otherwise not specified in this Recommendation.

4.1.5.1.1.7 If the *segmentation control* service parameter has requested that record boundaries be respected and there is no observable record structure to the file, an Invalid File Structure fault shall be declared.

NOTE – The algorithm for observing record structure in a file is not specified in this Recommendation. It is an implementation matter at the sender and is not relevant at the receiver.

4.1.5.1.1.8 When the Metadata PDU and all File Data PDUs have been issued (that is, when either (A) the file to be sent is bounded and the last octet of that file has been sent or (B) the file to be sent is unbounded and the transmission session for that file has concluded or (C) no file is to be sent),

a) the sending CFDP entity shall issue an EOF (No error) PDU;

b) the sending CFDP entity may optionally issue an `EOF-Sent.indication`, depending on a setting in the MIB;

c) the EOF (No error) PDU shall contain the checksum of the original file and the length of the file (which might not have been known at the time the Metadata PDU was issued);

d) the checksum shall be 32 bits in length and shall be calculated by the following method (see annex C for an example):

   1) it shall initially be set to all ‘zeroes’;

   2) it shall be calculated by modulo 2^{32} addition of all 4-octet words, aligned from the start of the file;

   3) each 4-octet word shall be constructed by copying some octet of file data, whose offset within the file is an integral multiple of 4 (such as 0, 4, 8, 12, etc.), into the first (high-order) octet of the word, and copying the next three octets of file data into the next three octets of the word;

   4) the results of the addition shall be carried into each available octet of the checksum unless the addition overflows the checksum length, in which case carry shall be discarded.

   NOTES

   1 In order to include in a checksum the content of a file data PDU whose offset is not an integral multiple of 4, it is necessary to align the data properly before adding 4-octet blocks of it to the checksum. Data at offset Q may be aligned by inserting N octets of value ‘zero’ before the first octet of the data, where N = Q mod 4 (the remainder obtained upon dividing Q by 4).

   2 In order to include in a checksum a sequence of M octets (the first of which is at a file offset that is an integral multiple of 4) where M is less than 4, it is

necessary to pad the data to length 4 before adding it to the checksum. The data may be padded by inserting (4 – M) octets of value 'zero' after the last octet of the data.

4.1.5.1.1.9 If the Transaction mode is unacknowledged, unacknowledged procedures shall be invoked at the receiving entity. If the Transaction mode is acknowledged, acknowledged procedures shall be invoked at the receiving entity.

4.1.5.1.1.10 The flow label may optionally be used to support prioritization and preemption schemes.

NOTE – The use of the flow label is implementation specific.

4.1.5.1.2 Copy File Procedures at the Receiving Entity

4.1.5.1.2.1 The receiving CFDP entity shall store the value of the transmission mode bit contained in the first received PDU of a transaction and use it for subsequent processing of the transaction. If the receiving CFDP entity is incapable of operating in this mode, an Invalid Transmission Mode fault shall be declared.

4.1.5.1.2.2 The receiving CFDP entity shall store fault handler overrides, file size, flow label, and file name information contained in the Metadata PDU and use it for subsequent processing of the transaction.

4.1.5.1.2.3 For transactions that deliver more than just metadata, the receiving CFDP entity shall establish whether the filestore can accept the file and, if it cannot, shall declare a Filestore Rejection fault.

4.1.5.1.2.4 At the receiver,

a) receipt of a Metadata PDU shall cause the receiving CFDP to issue a `Metadata-Recv.indication`:

   – if the Metadata PDU contains one or more messages to user, they shall be passed as a parameter of the `Metadata-Recv.indication`;

   – if the Metadata PDU contains filestore requests, these shall be stored for execution at the conclusion of delivery of the transaction's file data, if any;

b) received File Data PDUs shall be assembled into a file using the offset and segment size information of the File Data PDU:

   – any repeated data shall be discarded;

   – if the segmentation control flag contained in the Metadata PDU indicates that record boundaries have been respected, the alignment of records in the received File Data PDU may be used to assist in record reconstruction;

c) on receipt of the EOF (No error) PDU:

- a checksum shall be calculated for the delivered file;
- the calculated and received file checksums shall be compared;
- a File Checksum Failure fault shall be declared if the compared checksums are not equal.

NOTE – The action taken upon such error need not necessarily entail discarding the delivered file. The default handler for File Checksum Failure faults may be Ignore, causing the discrepancy to be announced to the user in a `Fault.indication` but permitting the completion of the Copy File procedure at the receiving entity. This configuration setting might be especially appropriate for transactions conducted in unacknowledged mode.

4.1.5.1.2.5 Upon receipt of the EOF (No error) PDU, the file size indicated in the PDU shall be compared to the transaction reception progress and a File Size Error fault declared if the progress exceeds the file size.

NOTE – The action taken upon such error need not necessarily entail discarding the delivered file. The default handler for File Size Error faults may be Ignore, causing the discrepancy to be announced to the user in a `Fault.indication` but permitting the completion of the Copy File procedure at the receiving entity. This configuration setting might be especially appropriate for transactions conducted in unacknowledged mode.

4.1.5.1.2.6 If the Transaction mode is unacknowledged, unacknowledged procedures shall be invoked at the receiving entity. If the Transaction mode is acknowledged, acknowledged procedures shall be invoked at the receiving entity.

4.1.5.1.2.7 The flow label may optionally be used to support prioritization and preemption schemes.

NOTE – The use of the flow label is implementation specific.

4.1.5.2 Optional Copy File Procedures at the Receiving Entity

Receipt of a File Data PDU may optionally cause the receiving CFDP to issue a `File-Segment-Recv.indication`.

4.1.5.3 Unacknowledged Mode Procedures

4.1.5.3.1 Unacknowledged Mode Procedures at the Sending Entity

Transmission of an EOF (No error) PDU shall cause the sending CFDP entity to issue a Notice of Completion (Completed).

4.1.5.3.2 Unacknowledged Mode Procedures at the Receiving Entity

Receipt of an EOF (No error) PDU shall cause the receiving CFDP entity to issue a Notice of Completion (Completed).

NOTE – Unacknowledged-mode transactions always terminate on receipt of the EOF (No error) PDU; therefore any Metadata or file data PDU received after the EOF (No error) PDU for the same transaction may be ignored.

4.1.5.4 Acknowledged Mode Procedures

4.1.5.4.1 Acknowledged Mode Procedures at the Sending Entity

4.1.5.4.1.1 The sending CFDP entity shall respond to all received NAK PDUs by retransmitting the requested Metadata PDU and/or the extents of the data file defined by the start and end offsets of the segment requests in the NAK PDU.

NOTE – The retransmission buffer for a File Copy procedure may be private to the protocol or (for greater efficiency in non-volatile storage space management) may be a publicly visible file. This is an implementation matter, as is the manner in which the buffer is released.

4.1.5.4.1.2 Segmentation of the retransmitted data shall not be constrained by the segmentation of the original transmission or by the start and end offsets in the NAK PDU. The constraints imposed by the segmentation control parameter, however, remain in effect.

4.1.5.4.1.3 Receipt of the Finished (No error) PDU with End System Status flag set to '1' shall cause the sending CFDP entity to issue a Notice of Completion (Completed).

4.1.5.4.1.4 Positive Acknowledgment procedures shall be applied to the EOF (No error) and Finished (No error) PDUs with the Expected Responses being ACK (EOF) PDU and ACK (Finished) PDU, respectively.

4.1.5.4.2 Acknowledged Mode Procedures at the Receiving Entity

4.1.5.4.2.1 Repeated file data or Metadata shall be discarded.

4.1.5.4.2.2 A file data gap shall be detected when any of the following conditions is true:

- the offset of the first octet of the first extent of received file data is not zero;
- the offset of the last octet of some contiguous extent of file data differs from the offset of the first octet of the next contiguous extent of data of the same file by more than one;
- the offset of the last octet of the last extent of file data differs from the length of the file as stated in the EOF (No error) PDU by more than one.

4.1.5.4.2.3 Lost Metadata shall be detected upon receipt of the first non-Metadata PDU (File Data or EOF (No error)) of a transaction for which no Metadata PDU has yet been received.

4.1.5.4.2.4 Detection of file data gaps or loss of Metadata shall cause the receiving CFDP entity to issue NAK PDUs as follows:

a) Each NAK shall identify the subset of file data to which it pertains, i.e., the *scope* of the NAK. The scope of a NAK is expressed as two offsets within the file, indicating the start and end of the scope.

b) In addition to its scope, each NAK shall contain zero or more segment requests. The segment request(s) in a NAK shall identify the start offsets and end offsets of all extents of file data within its scope which have not yet been received, and shall also identify missing metadata if any.

c) The start offsets and end offsets of segment requests do not need to relate in any way to the original segmentation of the file.

d) One *NAK sequence* shall be issued for each event upon which incremental gap reporting is deemed necessary by any Incremental Lost Segment Detection procedures that are in effect.

   - Each NAK sequence shall pertain to some subset of file data, i.e., the *scope* of the NAK sequence.
   - The start of scope for a NAK sequence shall be the end of scope of the previous NAK sequence issued for the same transaction, or zero if there has been no prior NAK sequence.
   - The end of scope for a NAK sequence shall be the current reception progress at the time of the event that caused issuance of the NAK sequence.
   - Each NAK sequence shall comprise as many NAK PDUs as are required, taking into consideration any limits on the maximum size of the content of a UT_SDU, to contain all the segment requests that must be sent in order to request retransmission of all the missing file data and Metadata within the scope of the NAK sequence.

   NOTE – Normally each NAK sequence will comprise only a single NAK.

– The start of scope for the first NAK in a sequence shall be the start of scope for the NAK sequence; the start of scope for every other NAK in the sequence shall be the end of scope of the prior NAK. The end of scope for the last NAK in the sequence shall be the end of scope for the NAK sequence itself; the end of scope for every other NAK in the sequence shall be the end offset of the NAK's last segment request.

e) The various Incremental Lost Segment Detection procedures are not mutually exclusive. Selection of incremental lost segment detection procedures to implement at a given entity is a local matter.

f) Upon initial receipt of the EOF (No error) PDU for a transaction, the Deferred Lost Segment Detection procedures shall be performed as follows: if all of the transaction's metadata and file data (if any) have been received, then transaction data reception shall be deemed Complete. Otherwise:

1) If any file data gaps or lost Metadata are detected for which no segment requests were included in previously issued NAK PDUs, the receiving CFDP entity shall issue a NAK sequence as described above.

2) A transaction-specific NAK timer shall be started. The timer shall have an implementation-specific expiry period. When the timer expires, the receiving entity shall determine whether or not any of the transaction's file data or metadata have yet to be received. If all of the transaction's metadata and file data (if any) have been received, then transaction data reception shall be deemed Complete. Otherwise, the receiving entity shall issue a NAK sequence whose scope begins at zero and extends through the entire length of the file, and the timer shall be reset.

NOTE – Because the expiry period of the NAK timer is wholly implementation-specific, it is not required to be the same on successive iterations of this data completeness determination cycle; in fact, it might even be dynamically altered (possibly multiple times) prior to any single expiration. In effect, re-examination of the file to check for data reception completeness – and potentially initiate issuance of a NAK sequence – may occur at any time following reception of the EOF (No error) PDU, at the sole discretion of the implementation.

g) An implementation-specific measure of NAK activity shall be maintained for each transaction, and, if an implementation-specific limit is reached for a given transaction, a NAK Limit Reached fault shall be declared.

NOTE – A typical implementation of NAK activity limit is a limit on the number of successive times the NAK timer is allowed to expire without intervening reception of file data and/or metadata that had not previously been received.

4.1.5.4.2.5 At the moment the transaction data reception is deemed Complete, as explained in 4.1.5.4.2.4 (f) above, the receiving CFDP entity shall:

a) issue a Notice of Completion (Completed);

b) issue a Finished (No error) PDU with End System Status flag set to '1'. Any filestore responses shall be carried as parameters of the Finished (No error) PDU.

4.1.5.4.2.6 Positive Acknowledgment procedures shall be applied to the EOF (No error) PDU and to any Finished (No error) PDU produced, with the Expected Responses being ACK (EOF) PDU and ACK (Finished) PDU respectively.

4.1.5.4.3 Incremental Lost Segment Detection Procedures at the Receiving Entity

4.1.5.4.3.1 General

None of the Incremental Lost Segment Detection procedures shall remain in effect, for a given transaction, after initial reception of the EOF (No error) PDU.

4.1.5.4.3.2 Immediate

In Immediate Mode, each file data gap or lost Metadata detected prior to reception of the EOF (No error) PDU shall cause the receiving CFDP entity to issue a NAK sequence immediately.

4.1.5.4.3.3 Prompted

In Prompted mode, upon receipt of a Prompt PDU prior to reception of the EOF (No error) PDU, the receiving CFDP entity shall issue a NAK sequence. If there are no missing file data or metadata, the NAK sequence shall comprise a single NAK containing no segment requests.

4.1.5.4.3.4 Asynchronous

In Asynchronous mode, in response to an implementation-specific external event occurring prior to reception of the EOF (No error) PDU, the receiving CFDP entity shall issue a NAK sequence. If there are no missing file data or metadata, the NAK sequence shall comprise a single NAK containing no segment requests.

NOTE – Examples of external events are the opening of a window of opportunity, for missions having sporadic contact, or a signal from a periodic timer at the receiving CFDP entity.

4.1.5.4.4 Incremental Lost Segment Detection Procedures at the Sending Entity

In response to an implementation-specific external event, the sending CFDP entity may issue a Prompt (NAK) PDU.

4.1.5.5 Keep Alive Procedures

4.1.5.5.1 Overview

The Keep Alive mechanism is provided as a means by which a sending entity may monitor the progress of file data reception at the receiving entity, so that problems may be detected prior to acknowledgment of the EOF (No error) PDU.

4.1.5.5.2 Keep Alive Procedures at the Receiving Entity

4.1.5.5.2.1 In all acknowledged modes, the receiving CFDP entity may periodically send a Keep Alive PDU to the sending CFDP entity reporting on the transaction's reception progress so far at this entity.

4.1.5.5.2.2 The Keep Alive PDU shall also be sent in response to receipt of a Prompt (Keep Alive) PDU.

4.1.5.5.2.3 However, transmission of Keep Alive PDUs shall cease upon receipt of the transaction's EOF (No error) PDU.

4.1.5.5.3 Keep Alive Procedures at the Sending Entity

4.1.5.5.3.1 At the sending CFDP entity, if the discrepancy between the reception progress reported by the Keep Alive PDU and the transaction's transmission progress so far at this entity exceeds a preset limit, the sending CFDP entity may optionally declare a Keep Alive Limit Reached fault.

4.1.5.5.3.2 In response to an implementation-specific stimulus, the sending Entity may generate a Prompt (Keep Alive) PDU to force the receiving entity to send a Keep Alive PDU.

4.1.5.6 Cancel Response Procedures

4.1.5.6.1 Cancel Response Procedures at the Receiving Entity

4.1.5.6.1.1 Receipt of an EOF (cancel) PDU shall cause the receiving CFDP entity to issue a Notice of Completion (Canceled).

4.1.5.6.1.2 If an acknowledged mode is in effect, Positive Acknowledgement procedures shall be applied to the EOF (cancel) PDU with the Expected Response being an ACK (EOF) PDU.

4.1.5.6.1.3 The incomplete data shall be either discarded or retained according to the option set in the MIB.

4.1.5.6.2 Cancel Response Procedures at the Sending Entity

4.1.5.6.2.1 Receipt of a Finished (cancel) PDU shall cause the sending CFDP entity to issue a Notice of Completion (Canceled).

4.1.5.6.2.2 If an acknowledged mode is in effect, Positive Acknowledgement procedures shall be applied to the Finished (cancel) PDU with the Expected Response being an ACK (Finished) PDU.

4.1.5.7 Resume Procedures

4.1.5.7.1 General

Resume procedures apply upon receipt of a `Resume.request` primitive submitted by the CFDP user. However:

a) a `Resume.request` primitive shall be ignored if it pertains to a transaction that is not currently suspended;

b) if the transaction to which a `Resume.request` primitive pertains is currently not only suspended but also frozen (as defined in 4.1.11), then the transaction shall be considered no longer suspended but the only applicable procedure shall be the issuance of a `Resumed.indication`.

4.1.5.7.2 Resume Procedures at the Sending Entity

4.1.5.7.2.1 On receipt of a `Resume.request` primitive, the sending CFDP entity shall

a) resume transmission of file segments;

b) issue a `Resumed.indication`.

4.1.5.7.2.2 If operating in acknowledged mode,

a) any suspended transmission of Prompt PDUs shall be resumed;

b) the application of Positive Acknowledgment Procedures to PDUs previously issued by this entity shall be resumed.

4.1.5.7.3 Resume Procedures at the Receiving Entity

4.1.5.7.3.1 On receipt of a `Resume.request` primitive, the receiving CFDP entity shall

a) resume transmission of NAK PDUs;

b) resume any suspended transmission of Keep Alive PDUs;

c) issue a `Resumed.indication`.

4.1.5.7.3.2 The application of Positive Acknowledgment Procedures to PDUs previously issued by this entity shall be resumed.

4.1.5.8 Report Procedures

Upon receipt of a `Report.request` primitive during a Copy File procedure, the CFDP entity shall issue a `Report.indication` primitive providing information on the progress of the transaction.

NOTES

1 Since there is no interaction with any other protocol entity, there is no protocol associated with this procedure.

2 The `Report.indication` primitive may also be issued on an asynchronous basis without the necessity for a `Report.request` primitive.

4.1.6 POSITIVE ACKNOWLEDGEMENT PROCEDURES

4.1.6.1 Positive Acknowledgement Procedures at PDU Sending End

If Positive Acknowledgement procedures apply to a PDU,

a) upon issuing the PDU the sending CFDP entity shall start a timer and retain the PDU for retransmission as necessary;

b) if the Expected Response is not received before expiry of the timer, the sending CFDP entity shall reissue the original PDU;

c) the sending CFDP entity shall keep a tally of the number of transmission retries;

d) if a preset limit is exceeded, the sending CFDP entity shall declare a Positive ACK Limit Reached fault;

e) receipt of the Expected Response shall cause the sending CFDP to release the retained PDU.

4.1.6.2 Positive Acknowledgement Procedures at PDU Receiving End

Receipt of a PDU to which Positive Acknowledgement procedures are applied shall cause the receiving CFDP entity immediately to issue the Expected Response.

NOTES

1 By issuing the Expected Response the CFDP entity **only** confirms receipt of the PDU; issuance of the Expected Response does not imply that the receiving CFDP entity has taken any other action as a result. For example, production of an ACK (EOF (Cancel)) PDU implies that an EOF (Cancel) PDU was received but not necessarily that the referenced transaction was canceled.

2 The receiving CFDP entity is **always** required to issue the Expected Response upon receipt of a PDU to which Positive Acknowledgment procedures are applied. The purpose of the Expected Response is to turn off the PDU retransmission timer at the sending end. This purpose must be served regardless of the status of the transaction: undefined, active, terminated, or unrecognized.

4.1.7 FAULT HANDLING PROCEDURES

4.1.7.1 Whenever a fault condition is detected and declared, the appropriate fault handler shall be invoked. A default handler shall be specified in the MIB for each type of fault condition, but for any particular transaction a fault handler override may have been stipulated for one or more types of fault condition; fault handler overrides, where present, shall be invoked in preference to the corresponding default fault handlers.

NOTE – Until the Metadata for a transaction is received, the receiving CFDP entity cannot know what fault handler overrides have been stipulated and therefore can only invoke the default fault handlers. Stipulating a fault handler override in a Put request therefore does not guarantee that the receiving entity will behave in the stipulated manner.

4.1.7.2 Invocation of a fault handler shall result in one of the following:

a) issuance of a Notice of Cancellation;

b) issuance of a Notice of Suspension, but only if the affected transaction was sent in ‘acknowledged’ transmission mode or the fault condition was declared at the source entity of the transaction;

c) no protocol action; i.e., the fault condition is ignored (note that this may result in unpredictable protocol behavior) and a Fault.indication with condition code identifying the fault condition is issued to the CFDP user;

d) abandonment of the transaction, resulting in an `Abandoned.indication` with condition code identifying the fault condition.

4.1.7.3 All faults shall be logged, even if no action is taken.

4.1.8 FILESTORE PROCEDURES

4.1.8.1 Filestore requests shall be executed only if any associated Copy File procedure proceeded to completion.

4.1.8.2 Filestore requests shall be transmitted in the Directive Parameter field of the Metadata PDU in the order in which they were submitted in the Put primitive (see 3.5.1).

4.1.8.3 Execution of filestore requests is mandatory. Filestore requests shall be executed in the order in which they are received in the Directive Parameter field of the Metadata PDU.

4.1.8.4 Execution of a filestore request shall result in the generation of a Filestore Responses parameter. If acknowledged mode is in effect, the Filestore Responses parameter shall be transmitted to the originator of the request via the Finished (No error) PDU.

4.1.8.5 If any filestore request obtained from a given Metadata PDU does not succeed, no subsequent filestore requests from the same Metadata PDU shall be executed; for each of these non-executed subsequent filestore requests, the filestore request status code returned in the resulting Filestore Responses parameter shall be 'not performed'.

4.1.8.6 Failure of a filestore request shall **not** result in the declaration of a fault of any kind.

4.1.9 INACTIVITY MONITOR PROCEDURES

4.1.9.1 For a particular transaction, if there is a cessation of PDU reception for a specified time period (the transaction inactivity limit) then an Inactivity fault condition shall be declared.

4.1.9.2 This requirement does not apply to the sending entity of an unacknowledged mode transfer.

4.1.9.3 The default handler for Inactivity faults shall be to Cancel the transaction.

4.1.10 INTERNAL PROCEDURES

4.1.10.1 Notice of Completion Procedures

NOTES

1 In the following subsections, the name of a Notice of Completion may include, in parentheses, the name of the specific disposition of the affected Copy File procedure, either 'Completed' indicating successful completion of the procedure or 'Canceled' indicating cancellation of the procedure. Disposition name may be omitted from the name of a Notice of Completion when it is irrelevant in the context in which the name is used.

2 It is possible for a Notice of Completion (Canceled) to be issued after a Notice of Completion (Completed) has been issued for the same transaction. This could be caused, for example, by the receiving entity's repeated failure to receive an ACK PDU in response to its Finished (No error) PDU, resulting in the declaration of a Positive ACK Limit Reached fault. The standard Notice of Completion Procedures continue to apply in this event. At the receiving entity the received file will already have been delivered to the user for normal processing by the time the Notice of Completion (Canceled) is issued.

4.1.10.1.1 Notice of Completion Procedures at the Sending Entity

4.1.10.1.1.1 On Notice of Completion of the Copy File procedure, the sending CFDP entity shall

a) release all unreleased portions of the file retransmission buffer;

b) stop transmission of file segments and metadata.

4.1.10.1.1.2 If sending in acknowledged mode,

a) any transmission of Prompt PDUs shall be terminated;

b) the application of Positive Acknowledgment Procedures to PDUs previously issued by this entity shall be terminated.

4.1.10.1.1.3 At the sending entity,

a) the CFDP entity shall issue a `Transaction-Finished.indication` primitive indicating the condition in which the transaction was completed;

b) if the file was sent in acknowledged mode and the condition in which the transaction was completed is 'No error' and the Finished PDU whose arrival completed the transaction contained a Filestore Responses parameter, then that Filestore Responses parameter shall be passed in the `Transaction-Finished.indication` primitive.

4.1.10.1.2 Notice of Completion Procedures at the Receiving Entity

4.1.10.1.2.1 If receiving in acknowledged mode:

a) transmission of NAK PDUs, whether in response to NAK timer expiration or in response to any other events, shall be terminated;

b) any transmission of Keep Alive PDUs shall be terminated;

c) the application of Positive Acknowledgment Procedures to PDUs previously issued by this entity shall be terminated.

4.1.10.1.2.2 In any case,

a) if the procedure disposition cited in the Notice of Completion is 'Completed', the receiving CFDP entity shall execute any filestore requests conveyed by the Put procedure;

b) the CFDP entity may optionally issue a `Transaction-Finished.indication` primitive indicating the condition in which the transaction was completed;

c) if available, a Status Report shall be passed as a parameter of the `Transaction-Finished.indication` primitive.

4.1.10.2 Notice of Cancellation Procedures

4.1.10.2.1 General

At any time during a Copy File procedure, either the sending CFDP entity or receiving CFDP entity may issue a Notice of Cancellation.

NOTE – The Notice of Cancellation may be issued in reaction to the declaration of a fault or to receipt of a `Cancel.request` primitive submitted by the CFDP user.

4.1.10.2.2 Notice of Cancellation Procedures at the Sending Entity

4.1.10.2.2.1 On Notice of Cancellation of the Copy File procedure, the sending CFDP entity shall

a) issue a Notice of Completion (Canceled);

b) issue an EOF (cancel) PDU indicating the reason for transaction termination: `Cancel.request` received or the condition code of the fault whose declaration triggered the Notice of Cancellation.

4.1.10.2.2.2 If sending in acknowledged mode,

a) Positive Acknowledgment procedures shall be applied to the EOF (cancel) PDU with the Expected Response being an ACK (EOF) PDU;

b) any PDU received after issuance of the EOF (cancel) PDU and prior to receipt of the Expected Response shall be ignored, except that all Positive Acknowledgment procedures remain in effect.

4.1.10.2.2.3 Any fault declared in the course of transferring the EOF (cancel) PDU must result in abandonment of the transaction.

4.1.10.2.3 Notice of Cancellation Procedures at the Receiving Entity

4.1.10.2.3.1 On Notice of Cancellation of the Copy File procedure, the receiving CFDP entity shall issue a Notice of Completion (Canceled).

4.1.10.2.3.2 The incomplete data shall be either discarded or retained according to the option set in the MIB.

NOTE – On Notice of Cancellation for a transaction for which a Notice of Completion (Completed) was previously declared, all file data have necessarily already been received and therefore there are no incomplete data to discard or retain.

4.1.10.2.3.3 If receiving in acknowledged mode,

a) the receiving CFDP entity shall issue a Finished (cancel) PDU indicating the reason for transaction termination: `Cancel.request` received or the condition code of the fault whose declaration triggered the Notice of Cancellation;

b) Positive Acknowledgment procedures shall be applied to the Finished (cancel) PDU with the Expected Response being an ACK (Finished) PDU;

c) any PDU received after issuance of the Finished (cancel) PDU and prior to receipt of the Expected Response shall be ignored, except that all Positive Acknowledgment procedures remain in effect;

d) any fault declared in the course of transferring this PDU must result in abandonment of the transaction.

4.1.10.3 Notice of Suspension Procedures

4.1.10.3.1 General

4.1.10.3.1.1 At any time during a Copy File procedure, either the sending CFDP entity or the receiving CFDP entity may issue a Notice of Suspension.

NOTE – A Notice of Suspension may be issued in reaction to the declaration of a fault or to receipt of a `Suspend.request` primitive submitted by the CFDP user.

4.1.10.3.1.2 However, a Notice of Suspension shall be ignored if it pertains to a transaction that is already suspended or if it is issued by the receiving CFDP entity for a transaction sent in Unacknowledged mode.

4.1.10.3.2 Notice of Suspension Procedures at the Sending Entity

4.1.10.3.2.1 On Notice of Suspension of the Copy File procedure, the sending CFDP entity shall

a) suspend transmission of file segments;

b) save the status of the transaction.

4.1.10.3.2.2 If operating in acknowledged mode,

a) any transmission of Prompt PDUs shall be suspended;

b) the application of Positive Acknowledgment Procedures to PDUs previously issued by this entity shall be suspended.

4.1.10.3.2.3 The sending entity shall issue a `Suspended.indication`.

4.1.10.3.3 Notice of Suspension Procedures at the Receiving Entity

On Notice of Suspension of the Put procedure, the receiving CFDP entity shall

a) suspend transmission of NAK PDUs;

b) suspend any transmission of Keep Alive PDUs;

c) suspend the application of Positive Acknowledgment Procedures to PDUs previously issued by this entity;

d) save the status of the transaction.

4.1.11 LINK STATE CHANGE PROCEDURES

4.1.11.1 General

Link state change procedures shall apply upon notification, by the Management Information Base (MIB), of the start and end of opportunities to transmit PDUs to, or receive PDUs from, specific remote CFDP entities.

NOTE – The start of a transmission opportunity nominally corresponds to some moment at which the remote entity will begin listening for PDUs from the local CFDP entity (e.g., will have pointed its antenna appropriately and applied power to its receiver), less the applicable one-way light time between the local and remote entities; similarly, the end of a transmission opportunity nominally precedes, by one-way light time, the moment at which the remote entity will cease listening for PDUs from the local entity. A reception opportunity likewise nominally begins at some moment at which the remote entity will begin sending PDUs to the local entity, plus one-way light time, and ends at the moment at which the remote entity will cease transmission, again plus one-way light time. However, the manner in which these notifications are initiated and delivered to CFDP is wholly a matter of implementation.

4.1.11.2 Transmission Opportunity Procedures

4.1.11.2.1 On notification of the end of an opportunity to transmit to a specified remote CFDP entity, the CFDP entity shall 'freeze' transmission for all transactions for which it is the sending entity and the specified remote entity is the receiving entity.

4.1.11.2.2 The freezing of transmission for a transaction shall have the same effects as suspension of that transaction by the sending entity (see 4.1.10.3.2), except that no `Suspended.indication` shall be issued and the transaction shall not be considered suspended.

4.1.11.2.3 On notification of the start of an opportunity to transmit to a specified remote CFDP entity, the CFDP entity shall 'thaw' transmission for all transactions for which it is the sending entity and the specified remote entity is the receiving entity.

4.1.11.2.4 The thawing of transmission for a transaction shall have the same effects as resumption of that transaction by the sending entity (see 4.1.5.7.2.1), except that (a) no `Resumed.indication` shall be issued and (b) thawing transmission for a suspended transaction shall have no effects whatsoever.

4.1.11.3 Reception Opportunity Procedures

4.1.11.3.1 On notification of the end of an opportunity to receive from a specified remote CFDP entity, the CFDP entity shall freeze reception for all transactions for which it is the receiving entity and the specified remote entity is the sending entity, except those that are in Unacknowledged mode.

4.1.11.3.2 The freezing of reception for a transaction shall have the same effects as suspension of that transaction by the receiving entity (see 4.1.10.3.3), except that no `Suspended.indication` shall be issued and the transaction shall not be considered suspended.

4.1.11.3.3 On notification of the start of an opportunity to receive from a specified remote CFDP entity, the CFDP entity shall thaw reception for all transactions for which it is the

receiving entity and the specified remote entity is the sending entity, except those that are in Unacknowledged mode.

4.1.11.3.4 The thawing of reception for a transaction shall have the same effects as resumption of that transaction by the receiving entity (see 4.1.5.7.3), except that (a) no `Resumed.indication` shall be issued and (b) thawing reception for a suspended transaction shall have no effects whatsoever.

5 PDU FORMATS

5.1 GENERAL

5.1.1 PDUs shall consist of a fixed-format PDU header followed by a fixed- or variable-format PDU data field.

5.1.2 The PDU header shall consist of the fields shown in table 5-1. Optional fields are present only when so indicated by the corresponding flag bit.

5.1.3 The PDU header fields shall be transmitted in the order of presentation in table 5-1.

NOTE – Every PDU is associated with a unique transaction by the source Entity ID together with the Transaction sequence number.

Table 5-1: Fixed PDU Header Fields

Field	Length (bits)	Values	Comment
Version	3	'000'	For the first version.
PDU type	1	'0' — File Directive '1' — File Data	
Direction	1	'0' — toward file receiver '1' — toward file sender	Used to perform PDU forwarding.
Transmission Mode	1	'0' — acknowledged '1' — unacknowledged	
CRC Flag	1	'0' — CRC not present '1' — CRC present	
Reserved for future use	1	set to '0'	
PDU Data field length	16		In octets.
CRC	16		Optional.
Reserved for future use	1	set to '0'	
Length of entity IDs	3		Number of octets in entity ID less one; i.e., '0' means that entity ID is one octet. Applies to all entity IDs in the PDU header.
Reserved for future use	1	set to '0'	
Length of Transaction sequence number	3		Number of octets in sequence number less one; i.e., '0' means that sequence number is one octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction.
Transaction sequence number	variable		Uniquely identifies the transaction, among all transactions originated by this entity.
Destination entity ID	variable		Uniquely identifies the entity that is the final destination of the transaction's metadata and file data.

5.1.4 The sequence number shall be sequentially allocated by the CFDP Entity responding to the `Put.request`.

5.1.5 Two PDU types are defined:

– File Directive;

– File Data.

5.1.6 The PDU Data field can contain data items in any one of four formats:

– fixed-format data;

– Length, Value (LV) format;

– Type, Length, Value (TLV) format;

– variable-length file data.

NOTES

1 Fixed-format data is precisely defined in this document and is optimized for efficiency of the protocol where the format of the data is not required to be variable. Fixed format data fields are fixed in their position in the PDU and in their length.

2 Each PDU field referred to as an offset is encoded as the number of octets in the file prior to the octet described by that field; e.g., the offset of the first octet of the file is encoded as all 'zeroes'.

3 Entity IDs may be of different lengths in different PDUs, as may transaction sequence numbers, but these differences in length have no semantic significance. Two entity IDs of different lengths are compared by, in effect, inserting leading 'zeroes' to pad the shorter entity ID to the same length as the longer entity ID and then comparing the resulting values. The same procedure applies to comparisons between transaction sequence numbers of different lengths.

5.1.7 The general format of LV objects shall be as shown in table 5-2.

Table 5-2: LV Object Format

Field	Size (bits)	Values	Comment
Length	8	0 to 255	Length of value.
Value	8 × Length		

NOTE – LV format data is variable in its length but is invariable in its position in the PDU. It can, by virtue of its length field, be parsed by the protocol.

5.1.8 The format of a TLV object shall be as shown in table 5-3.

Table 5-3: TLV Object Format

Field	Size (bits)	Values	Comment
Type	8	See 5.4 below	Nature of value.
Length	8	0 to 255	Length of value.
Value	8 × Length		

NOTES

1 TLV format data is variable both in its length and in its position in the PDU. It can, however, by virtue of its type and length fields, be parsed by the protocol.

2 In the following subsections, parameters are characterized as either fixed parameters (indicated by a specific field length), LV parameters (indicated by the notation 'LV'), or TLV parameters (indicated by the notation 'TLV hh', where hh is the hexadecimal value in the Type field of the TLV object).

5.2 FILE DIRECTIVE PDUs

5.2.1 GENERAL

5.2.1.1 The data field of File Directives shall consist of a Directive Code octet followed by a Directive Parameter field.

5.2.1.2 The File Directive Code shall indicate one of the actions shown in table 5-4.

Table 5-4: File Directive Codes

Directive Code (hexadecimal)	Action
00	Reserved
01	Reserved
02	Reserved
03	Reserved
04	EOF PDU
05	Finished PDU
06	ACK PDU
07	Metadata PDU
08	NAK PDU
09	Prompt PDU
0C	Keep Alive PDU
0D–FF	Reserved

NOTE – The Directive Parameter fields for the above actions are described, in order of transmission, in the following subsections.

5.2.1.3 In several cases, the Directive Parameter field of a File Directive includes a four-bit Condition Code. The Condition Code shall in each case indicate one of the conditions shown in table 5-5.

Table 5-5: Condition Codes

Condition Code (binary)	Condition
0000	No error
0001	Positive ACK limit reached
0010	Keep alive limit reached
0011	Invalid transmission mode
0100	Filestore rejection
0101	File checksum failure
0110	File size error
0111	NAK limit reached
1000	Inactivity detected
1001	Invalid file structure
1010 – 1101	(reserved)
1110	`Suspend.request` received
1111	`Cancel.request` received

5.2.2 END-OF-FILE PDU

The contents of the Parameter field for a File Directive having a Code of EOF PDU shall be as shown in table 5-6.

Table 5-6: End-of-File PDU Contents

Parameter	Length (bits)	Values	Comments
Condition Code	4	See table 5-5.	
Spare	4		
File Checksum	32		Modulo 2^{32} word-wide addition (where 'word' is defined as 4 octets) of all file segment data transmitted by the sender (regardless of the condition code, i.e., even if the condition code is other than 'No error'), aligned with reference to the start of file. For method, see 4.1.5.1.1.8.
File size	32		In octets. This value shall be the total number of file data octets transmitted by the sender, regardless of the condition code (i.e., it shall be supplied even if the condition code is other than 'No error').

5.2.3 FINISHED PDU

The contents of the Parameter field for a File Directive having a Code of Finished PDU shall be as shown in table 5-7.

Table 5-7: Finished PDU Contents

Parameter	Length (bits)	Values	Comments
Condition Code	4	See table 5-5.	
End System Status	1	'1' — Generated by End System.	
Delivery Code	1	'0' — Data Complete. '1' — Data Incomplete.	
Spare	2		
Filestore Responses	TLVs	See table 5-17.	A filestore response TLV must be included for each filestore request TLV of the Metadata PDU.

5.2.4 ACK PDU

The contents of the Parameter field for a File Directive having a Code of ACK PDU shall be as shown in table 5-8. The possible values of the Transaction Status parameter are as follows (all values in binary notation):

00 – Undefined. The transaction to which the acknowledged PDU belongs is not currently active at this entity, and the CFDP implementation **does not** retain transaction history. The transaction might be one that was formerly active and has been terminated, or it might be one that has never been active at this entity.

01 – Active. The transaction to which the acknowledged PDU belongs is currently active at this entity.

10 – Terminated. The transaction to which the acknowledged PDU belongs is not currently active at this entity, the CFDP implementation **does** retain transaction history, and the transaction is thereby known to be one that was formerly active and has been terminated.

11 – Unrecognized. The transaction to which the acknowledged PDU belongs is not currently active at this entity, the CFDP implementation **does** retain transaction history, and the transaction is thereby known to be one that has never been active at this entity.

Table 5-8: ACK PDU Contents

Parameter	Length (bits)	Values	Comments
Directive code	4	See table 5-4. Only EOF and Finished PDUs are acknowledged.	Directive code of the acknowledged PDU.
Directive subtype code	4		Binary 0000 for ACKs of all file directives.
Condition code	4	See table 5-5 .	Condition code of the acknowledged PDU.
Delivery code	2	For ACK of EOF PDU: binary 10 if Data Complete, binary 11 if Data Incomplete. Binary 00 for ACKs of all other file directives.	'Data Complete' means that all file data and Metadata have been received.
Transaction status	2		Status of the transaction in the context of the entity that is issuing the acknowledgment.

5.2.5 METADATA PDU

The contents of the Parameter field for a File Directive having a Code of Metadata PDU shall be as shown in table 5-9.

Table 5-9: Metadata PDU Contents

Parameter	Length (bits)	Values	Comments
Segmentation control	1	'0' — Record boundaries respected. '1' — Record boundaries not respected.	
Reserved for future use	7		Set to all 'zeroes'.
File size	32	Length of File. Set to all 'zeroes' for a file of unbounded size.	In octets.
Source file name	LV	LV Length field indicates zero length and LV value field omitted when there is no associated file, e.g., messages used for Proxy operations.	
Destination file name	LV	LV Length field indicates zero length and LV value field omitted when there is no associated file, e.g., messages used for Proxy operations.	
Options	TLVs		Filestore requests. Messages to user. Fault Handler overrides. Flow Label.

5.2.6 NAK PDU

5.2.6.1 The contents of the Parameter field for a File Directive having a Code of NAK PDU shall be as shown in table 5-10.

Table 5-10: NAK PDU Contents

Parameter	Length (bits)	Values	Comments
Start of scope	32	See 4.1.5.4.2.4.	
End of scope	32	See 4.1.5.4.2.4.	
Segment Requests (× N)	64 × N	See following table.	

5.2.6.2 Each Segment Request shall be as shown in table 5-11.

Table 5-11: Segment Request Form

Parameter	Length (bits)	Values	Comments
Start offset	32	Data — Offset of start of requested segment Metadata — 00000000 (hex)	In octets.
End Offset	32	Data — Offset of first octet after end of requested segment Metadata — 00000000 (hex)	In octets.

5.2.7 PROMPT PDU

The contents of the Parameter field for a File Directive having a Code of Prompt PDU shall be as shown in table 5-12.

Table 5-12: Prompt PDU Contents

Parameter	Length (bits)	Values	Comments
Response required	1	'0' — NAK '1' — Keep Alive	
Spare	7		

5.2.8 KEEP ALIVE PDU

The contents of the Parameter field for a File Directive having a Code of Keep Alive PDU shall be as shown in table 5-13.

Table 5-13: Keep Alive PDU Contents

Parameter	Length (bits)	Values	Comments
Progress	32	00000000-FFFFFFFF (hex)	In octets.

5.3 FILE DATA PDU

The contents of the data field of a File Data PDU shall be as shown in table 5-14.

Table 5-14: File Data PDU Contents

Parameter	Length (bits)	Values	Comments
Offset	32		In octets.
File data	variable		

5.4 TLV PARAMETERS

5.4.1 FILESTORE REQUEST TLV

5.4.1.1 The Type of the Filestore Request TLV shall be 00 hex.

5.4.1.2 The Value of the Filestore Request TLV shall be as shown in table 5-15.

Table 5-15: Filestore Request TLV Contents

Parameter	Length (bits)	Values	Comments
Action Code	4	See table 5-16.	
Spare	4		
First File Name	LV		
Second File Name	LV		Only present for some action codes.

5.4.1.3 The Action codes and their parameters are as shown in table 5-16.

Table 5-16: Filestore Request TLV Action Codes

Action Code	Action	Second File Name Present
'0000'	Create File	N
'0001'	Delete File	N
'0010'	Rename File	Y
'0011'	Append File	Y
'0100'	Replace File	Y
'0101'	Create Directory	N
'0110'	Remove Directory	N
'0111'	Deny File (delete if present)	N

5.4.2 FILESTORE RESPONSE TLV

5.4.2.1 The Type of the Filestore Response TLV shall be 01 hex.

5.4.2.2 The Value of the Filestore Response TLV shall be as shown in table 5-17.

Table 5-17: Filestore Response TLV Contents

Parameter	Length (bits)	Values	Comments
Action Code	4		As for Filestore Request TLV.
Status Code	4	See table 5-18.	
First File name	LV		
Second File Name	LV		Only present for some action codes, as for Filestore Request TLV.
Filestore Message	LV		Error message for a failed action; optional.

Table 5-18: Filestore Response Status Codes

Action Code	Action	Use of Filenames	Status Codes
'0000'	Create	First — filename to be created	'0000' — Successful '0001' — Create not allowed '1111' — Not performed
'0001'	Delete	First — filename to be deleted	'0000' — Successful '0001' — File does not exist '0010' — Delete not allowed '1111' — Not performed
'0010'	Rename	First — old filename Second — new filename	'0000' — Successful '0001' — Old File Name does not exist '0010' — New File Name already exists '0011' — Rename not allowed '1111' — Not performed
'0011'	Append	First — name of file whose contents form first part of new file and name of the new file. Second — name of file whose contents will form second part of new file.	'0000' — Successful '0001' — File Name 1 does not exist '0010' — File Name 2 does not exist '0011' — Append not allowed '1111' — Not performed
'0100'	Replace	First — filename whose contents are to be replaced Second — filename whose contents will replace the contents of the first filename	'0000' — Successful '0001' — File Name 1 does not exist '0010' — File Name 2 does not exist '0011' — Replace not allowed '1111' — Not performed
'0101'	Create Directory	First — name of directory to be created	'0000' — Successful '0001' — Directory cannot be created '1111' — Not performed
'0110'	Remove Directory	First — name of directory to be deleted	'0000' — Successful '0001' — Directory does not exist '0010' — Delete not allowed '1111' — Not performed
'0111'	Deny File	First — filename to be deleted	'0000' — Successful '0010' — Delete not allowed '1111' — Not performed

5.4.3 MESSAGE TO USER TLV

The Type of the Message to User TLV shall be 02 hex; the format of the Value (the message) is not defined.

NOTE – To enable interoperability it will typically be necessary for each message to contain some indication of its nature. The message identifier for the standard CFDP Proxy and Directory Operations messages (see section 6 below) is the presence of the ASCII characters 'cfdp' in the first four octets of each message; that message identification bit pattern is reserved for this purpose.

5.4.4 FAULT HANDLER OVERRIDE TLV

The Type of the Fault Handler Override TLV shall be 04 hex; the one-byte Value[1] shall be encoded as shown in table 5-19.

Table 5-19: Fault Handler Override Field Encoding

Parameter	Length (bits)	Values	Comments
Condition code	4	See table 5-5.	The condition codes for file completion, suspension, and cancellation requests are not applicable; these are not faults.
Handler code	4	'0000' — reserved for future expansion '0001' — issue Notice of Cancellation '0010' — issue Notice of Suspension '0011' — Ignore error '0100' — Abandon transaction '0101'–'1111' — reserved	

NOTE – If the Metadata PDU is not the first received PDU of a transaction, the fault handler settings in the MIB will be applied until the arrival of the Metadata PDU.

5.4.5 FLOW LABEL TLV

The Type of the Flow Label TLV shall be 05 hex; the format of the Value is not defined.

[1] The Value field of this TLV may be extended in future versions of this specification.

6 USER OPERATIONS

NOTE – This section describes standard interoperable procedures which use the services provided by the CFDP protocol.

6.1 RESERVED CFDP MESSAGE FORMAT

6.1.1 CFDP User Operations shall be accomplished through the exchange of Reserved CFDP Messages between CFDP users. These messages shall be transported as the values of Message to User TLV parameters in the metadata of standard CFDP transactions.

6.1.2 Each such message shall consist of a Reserved CFDP Message header formatted according to table 6-1, followed by message content whose format varies with the type of the message.

Table 6-1: Reserved CFDP Message Header

Field	Size (bits)	Values	Comments
Message identifier	32	The ASCII characters 'cfdp'.	Distinguishes Reserved CFDP Messages from all other user messages.
Message type	8	See following tables.	

6.1.3 The fields in the content of a Reserved CFDP Message shall contain data items as described in the CFDP Protocol Specification.

6.1.4 The fields shall be in fixed format, in variable format, or in LV format as defined in 5.1 above.

6.1.5 The **Originating Transaction ID** message is common to all User operations. Its message type shall contain the hexadecimal value 0A and its content shall be constructed as indicated in table 6-2.

Table 6-2: Originating Transaction ID Message

Field	Size (bits)	Values	Comments
Reserved for future use	1	set to '0'	
Length of entity ID	3		Number of octets in source entity ID less one; i.e., '0' means that source entity ID is one octet.
Reserved for future use	1	set to '0'	
Length of Transaction sequence number	3		Number of octets in sequence number less one; i.e., '0' means that sequence number is one octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction.
Transaction sequence number	variable		Uniquely identifies the transaction, among all transactions originated by this entity.

6.2 PROXY OPERATION

NOTE – The term 'proxy operation' refers to the use of CFDP services by some CFDP user, referred to as the 'originator' of the operation, to initiate the transmission of a file by some remote CFDP entity's user, referred to as the 'respondent', to some other entity's user, referred to as the 'beneficiary'. The beneficiary of a proxy operation might be either the originator itself (in which case the proxy operation functions as a 'Get') or some third CFDP entity's user.

6.2.1 GENERAL

To enable interoperability, the following mandatory behavior shall be observed by CFDP users that are in compliance with the CFDP proxy operations specification.

6.2.2 PROXY OPERATIONS MESSAGE TYPES

The message type field for each Reserved CFDP Message used in Proxy operations shall contain one of the values specified in table 6-3.

Table 6-3: Proxy Operations Message Types

Message Type (hexadecimal)	Interpretation
00	Remote Put Order
01	Remote Message to User
02	Remote Filestore Request
03	Remote Fault Handler Override
04	Remote Transmission Mode
05	Remote Flow Label
06	Remote Segmentation Control
07	Remote Put Finished
08	Remote Filestore Response
09	Remote Put Cancel

6.2.3 INITIATING A PROXY OPERATION

6.2.3.1 General

6.2.3.1.1 When required to initiate a proxy operation, the CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit whose metadata contains one or more of the Reserved CFDP Messages, defined in 6.2.3.2 through 6.2.3.8, which correspond to the parameters of the `Put.request` primitive (3.5.1).

6.2.3.1.2 The CFDP user that is the destination of the file delivery unit (the respondent) shall in turn use the contents of these messages to formulate a second `Put.request` primitive that accomplishes the desired remote file delivery.

6.2.3.1.3 The transaction ID assigned to the original request, as reported to the originating CFDP user by the corresponding `Transaction.indication` primitive, shall be cited in the transaction that constitutes the proxy's eventual report on the results of the requested proxy operation.

6.2.3.2 Remote Put Order

The Remote Put Order message is mandatory and shall be constructed as indicated in table 6-4.

Table 6-4: Remote Put Order Message

Field	Size (bits)	Values	Comments
Destination entity ID	Variable	LV	The ID of the beneficiary's CFDP entity.
Source file name	Variable	LV	Length is zero if parameter is omitted.
Destination file name	Variable	LV	Length is zero if parameter is omitted.

6.2.3.3 Remote Message to User

One or more Remote Message to User messages may be optionally included and shall be constructed as indicated in table 6-5.

Table 6-5: Remote Message to User Message

Field	Size (bits)	Values	Comments
Message text	Variable	LV	See 5.4.3.

6.2.3.4 Remote Filestore Request

One or more Remote Filestore Request messages may be optionally included and shall be constructed as indicated in table 6-6.

Table 6-6: Remote Filestore Request Message

Field	Size (bits)	Values	Comments
Length (octets)	8	Number of octets in the request field.	
Request	8 × length	The value of this field is a single CFDP filestore request as defined in table 5-17, above.	

6.2.3.5 Remote Fault Handler Override

One or more Remote Fault Handler Override messages may optionally be included and shall be constructed as indicated in table 6-7.

Table 6-7: Remote Fault Handler Override Message

Field	Size (bits)	Values	Comments
Fault handler override	8	See 5.4.4.	

6.2.3.6 Remote Transmission Mode

The Remote Transmission Mode message is optional and shall be constructed as indicated in table 6-8.

Table 6-8: Remote Transmission Mode Message

Field	Size (bits)	Values	Comments
Spare	7	All ‘zeroes’.	
Transmission mode	1	See table 5-1.	

6.2.3.7 Remote Flow Label

A Remote Flow Label message is optional and shall be constructed as indicated in table 6-9.

Table 6-9: Remote Flow Label Message

Field	Size (bits)	Values	Comments
Flow label	Variable	LV	See 5.4.5.

6.2.3.8 Remote Segmentation Control

A Remote Segmentation Control message is optional and shall be constructed as indicated in table 6-10.

Table 6-10: Remote Segmentation Control Message

Field	Size (bits)	Values	Comments
Spare	7	All 'zeroes'.	
Segmentation control	1	See table 5-9.	

6.2.4 RESPONDING TO A REMOTE PUT ORDER

Upon receipt of a Remote Put Order message, the respondent CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit with the following parameter(s):

- destination entity ID as specified in the Remote Put Order message;
- source file name as specified in the Remote Put Order message (if any);
- destination file name as specified in the Remote Put Order message (if any);
- segmentation control as specified in the Remote Segmentation Control message that was received in the same `Metadata-recv.indication` (if any);
- transmission mode as specified in the Remote Transmission Mode message that was received in the same `Metadata-recv.indication` (if any);
- flow label as specified in the Remote Flow Label message that was received in the same `Metadata-recv.indication` (if any);
- one fault handler override for each Remote Fault Handler message that was received in the same `Metadata-recv.indication`;

– one Message to User for each Remote Message to User message that was received in the same `Metadata-recv.indication`;

– one filestore request for each Remote Filestore Request message that was received in the same `Metadata-recv.indication`.

6.2.5 REPORTING ON THE RESULTS OF A REMOTE PUT ORDER

6.2.5.1 General

6.2.5.1.1 Upon receipt of a `Transaction-Finished.indication` indicating the completion of a transaction that was initiated in response to a Remote Put Order message, the respondent CFDP user that requested that transaction shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit (to the originator of the proxy operation) whose metadata contains one each of the Reserved CFDP Messages of message types Remote Put Finished (defined in 6.2.5.2) and Originating Transaction ID, together with zero of more Reserved CFDP Messages of message type Remote Filestore Response (defined in 6.2.5.3).

6.2.5.1.2 If for any reason the `Put.request` cannot be honored, the respondent CFDP user shall take no further action with regard to this proxy operation.

6.2.5.2 Remote Put Finished

The Remote Put Finished message is mandatory and shall be constructed as indicated in table 6-11.

Table 6-11: Remote Put Finished Message

Field	Size (bits)	Values	Comment
Condition code	4	See table 5-5.	
Spare	4	All ‘zeroes’.	

6.2.5.3 Remote Filestore Response

If the `Transaction-Finished.indication` included filestore responses, then for each such response one Remote Filestore Response message shall be included and shall be constructed as indicated in table 6-12.

Table 6-12: Remote Filestore Response Message

Field	Size (bits)	Values	Comments
Length	8	Number of octets in the response field.	
Response	8 *length	The value of this field is a single CFDP filestore response as defined in table 5-17, above.	

6.2.6 CANCELING A REMOTE PUT ORDER

6.2.6.1 General

When required to cancel a proxy operation, the proxy operation originator shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit whose metadata contains one each of the Reserved CFDP Messages of message types Remote Put Cancel, defined in 6.2.6.2, and Originating Transaction ID.

6.2.6.2 Remote Put Cancel

A Remote Put Cancel message is mandatory. It has no content.

6.2.7 RESPONDING TO A REMOTE PUT CANCEL

Upon receipt of a Remote Put Cancel message, the respondent CFDP user shall use the `Cancel.request` primitive to request cancellation of the transaction that was initiated in response to the one identified by the accompanying Originating Transaction ID message. The results of this action shall be reported according to 6.2.5.

6.3 DIRECTORY OPERATIONS

6.3.1 GENERAL

To enable interoperability, the following mandatory behavior shall be observed by CFDP users that are in compliance with the CFDP directory operations specification.

6.3.2 DIRECTORY OPERATIONS MESSAGE TYPES

The message type field for each Reserved CFDP Message used in Directory operations shall contain one of the values specified in table 6-13.

Table 6-13: Directory Operations Message Types

Message Type (hexadecimal)	Interpretation
10	Directory Listing Request
11	Directory Listing Response

6.3.3 INITIATING A DIRECTORY LISTING

6.3.3.1 General

When required to initiate a directory listing operation, the CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit whose metadata contains a Reserved CFDP Message of message type Directory Listing Request, defined in 6.3.3.2.

6.3.3.2 Directory Listing Request

The Directory Listing Request message shall be constructed as indicated in table 6-14.

Table 6-14: Directory Listing Request Message

Field	Size (bits)	Values	Comments
Directory Name	Variable	LV	
Directory File Name	Variable	LV	The file name and path at the filestore local to the requesting CFDP user in which the responding CFDP user should put the directory listing.

6.3.4 RESPONDING TO A DIRECTORY LISTING REQUEST

6.3.4.1 General

6.3.4.1.1 Upon receipt of a Directory Listing Request message, the remote CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit to the entity which sent the original Directory Listing Request message; the file data of this file delivery unit shall be a file that contains the directory listing, while the metadata of this file delivery unit shall contain one each of the Reserved CFDP Messages of message types Directory Listing Response (defined in 6.3.4.2) and Originating Transaction ID.

6.3.4.1.2 If the remote CFDP user is unable to provide a directory listing file, then the responding file delivery unit shall contain no file and the listing response code in the Directory Listing Response message shall indicate that the directory listing request was unsuccessful.

6.3.4.2 Directory Listing Response

The Directory Listing Response message shall be constructed as indicated in table 6-15.

Table 6-15: Directory Listing Response Message

Field	Size (bits)	Values	Comments
Listing Response Code	8	00h to 7Fh — successful 80h to FFh — unsuccessful	
Directory Name	Variable	LV	The name of the directory being listed, taken from the directory listing request.
Directory File Name	Variable	LV	The file name and path at the filestore local to the requesting CFDP in which the listing has been put, taken from the directory listing request.

6.3.5 RETRIEVING A DIRECTORY LISTING

Upon receiving the Reserved CFDP Message with a Directory Listing Response message type, the requesting CFDP user may retrieve the listing from the file specified in the message.

6.4 REMOTE STATUS REPORT OPERATIONS

6.4.1 GENERAL

To enable interoperability, the following mandatory behavior shall be observed by CFDP users that are in compliance with the CFDP remote status report operations specification.

6.4.2 REMOTE STATUS REPORT OPERATIONS MESSAGE TYPES

The Remote Status Report Message to User message type field shall contain one of the values specified in table 6-16.

Table 6-16: Remote Status Report Operations Message Types

Message Type (hexadecimal)	Interpretation
20	Remote Status Report Request
21	Remote Status Report Response

6.4.3 INITIATING A REMOTE STATUS REPORT

6.4.3.1 General

When required to initiate a remote status report operation, the CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit whose metadata contains the Reserved CFDP Message defined in 6.4.3.2.

6.4.3.2 Remote Status Report Request

The Remote Status Report Request message shall be constructed as indicated in table 6-17.

Table 6-17: Remote Status Report Request Message

Field	Size (bits)	Values	Comments
Reserved for future use	1	set to '0'	
Length of entity ID	3		Number of octets in source entity ID less one; i.e., '0' means that source entity ID is one octet.
Reserved for future use	1	set to '0'	
Length of Transaction sequence number	3		Number of octets in sequence number less one; i.e., '0' means that sequence number is one octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction for which a status report is requested.
Transaction sequence number	variable		Uniquely identifies the transaction, among all transactions originated by this entity.
Report File Name	Variable	LV	The file name and path at the filestore local to the requesting CFDP user in which the responding CFDP user should put the status report.

6.4.4 RESPONDING TO A REMOTE STATUS REPORT REQUEST

6.4.4.1 General

6.4.4.1.1 Upon receipt of a Remote Status Report Request message, the remote CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit to the entity which sent the original Status Report Request message; the file data of this file delivery unit shall be a file that contains the status report for the indicated transaction, while the metadata of this file delivery unit shall contain one each of the Reserved CFDP Messages of message types Remote Status Report Response (defined in 6.4.4.2) and Originating Transaction ID.

6.4.4.1.2 If the remote CFDP user is unable to provide a status report file, then the responding file delivery unit shall contain no file and the report response code in the Remote Status Report Response message shall indicate that the remote status report request was unsuccessful.

6.4.4.1.3 If the remote CFDP user is able to provide a status report file, the nature of the status report shall be as described in 3.3.8.

6.4.4.2 Remote Status Report Response

The Remote Status Report Response message shall be constructed as indicated in table 6-18.

Table 6-18: Remote Status Report Response Message

Field	Size (bits)	Values	Comments
Transaction status	2	Same values as defined for transaction status in the ACK PDU (see table 5-8).	
Spare	6	All 'zeroes'.	
Spare	1	'0'	This field and the following five fields identify the transaction whose status is being reported; they are taken from the status report request.
Length of Entity ID	3		
Spare	1	'0'	
Length of Sequence Number	3		
Source Entity ID	Variable		
Transaction Sequence Number	Variable		
Directory File Name	Variable	LV	The file name and path at the filestore local to the requesting CFDP in which the report has been put, taken from the status report request.

6.4.5 RETRIEVING A REMOTE STATUS REPORT

Upon receiving the Reserved CFDP Message with a Remote Status Report Response message type, the requesting CFDP user may retrieve the report from the file specified in the message.

6.5 REMOTE SUSPEND OPERATIONS

6.5.1 GENERAL

To enable interoperability, the following mandatory behavior shall be observed by CFDP users that are in compliance with the CFDP remote suspend operations specification.

6.5.2 REMOTE SUSPEND OPERATIONS MESSAGE TYPES

The Remote Suspend Message to User message type field shall contain one of the values specified in table 6-19.

Table 6-19: Remote Suspend Operations Message Types

Message Type (hexadecimal)	Interpretation
30	Remote Suspend Request
31	Remote Suspend Response

6.5.3 INITIATING A REMOTE SUSPEND

6.5.3.1 General

When required to initiate a remote suspend operation, the CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit whose metadata contains the Reserved CFDP Message defined in 6.5.3.2.

6.5.3.2 Remote Suspend Request

The Remote Suspend Request message shall be constructed as indicated in table 6-20.

Table 6-20: Remote Suspend Request Message

Field	Size (bits)	Values	Comments
Reserved for future use	1	set to '0'	
Length of entity ID	3		Number of octets in source entity ID less one; i.e., '0' means that source entity ID is one octet.
Reserved for future use	1	set to '0'	
Length of Transaction sequence number	3		Number of octets in sequence number less one; i.e., '0' means that sequence number is one octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction that is to be suspended.
Transaction sequence number	variable		Uniquely identifies the transaction, among all transactions originated by this entity.

6.5.4 RESPONDING TO A REMOTE SUSPEND REQUEST

6.5.4.1 General

6.5.4.1.1 Upon receipt of a Remote Suspend Request message, the remote CFDP user shall assure the suspension of the indicated transaction if possible; the CFDP `Suspend.request` primitive will nominally be used for this purpose. It shall then use the CFDP `Put.request` primitive to request delivery of a file delivery unit to the entity which sent the original Suspend Request message. The metadata of this file delivery unit shall contain one each of the Reserved CFDP Messages of message types Remote Suspend Response (defined in 6.5.4.2) and Originating Transaction ID.

6.5.4.1.2 It is possible for there to be multiple concurrent motivations for suspending a transaction; for example, a CFDP entity might receive a Remote Suspend Request for a transaction which is already suspended due to a locally detected fault condition. The CFDP user application is responsible for managing a queue of such suspension orders in such a way that a suspended transaction remains suspended until all such orders have been explicitly reversed by corresponding resume orders; in particular, a transaction for which a suspension-inducing fault condition was detected must remain suspended until the CFDP user

application determines that this fault condition no longer holds, and a transaction for which a remote suspend request was received from a given CFDP entity's user must remain suspended until a remote resume request for that transaction is received from the same user. The manner in which this is accomplished is an implementation matter.

6.5.4.2 Remote Suspend Response

The Remote Suspend Response message shall be constructed as indicated in table 6-21.

Table 6-21: Remote Suspend Response Message

Field	Size (bits)	Values	Comments
Suspension indicator	1	'0' — not suspended. '1' — suspended.	
Transaction status	2	Same values as defined for transaction status in the ACK PDU (see table 5-8).	
Spare	5	All 'zeroes'.	
Spare	1	'0'	This field and the following five fields identify the transaction whose suspension was requested; they are taken from the suspend request.
Length of Entity ID	3		
Spare	1	'0'	
Length of Sequence Number	3		
Source Entity ID	Variable		
Transaction Sequence Number	Variable		

6.6 REMOTE RESUME OPERATIONS

6.6.1 GENERAL

To enable interoperability, the following mandatory behavior shall be observed by CFDP users that are in compliance with the CFDP remote resume operations specification.

6.6.2 REMOTE RESUME OPERATIONS MESSAGE TYPES

The Remote Resume Message to User message type field shall contain one of the values specified in table 6-22.

Table 6-22: Remote Resume Operations Message Types

Message Type (hexadecimal)	Interpretation
38	Remote Resume Request
39	Remote Resume Response

6.6.3 INITIATING A REMOTE RESUME

6.6.3.1 General

When required to initiate a remote resume operation, the CFDP user shall use the CFDP `Put.request` primitive to request delivery of a file delivery unit whose metadata contains the Reserved CFDP Message defined in 6.6.3.2.

6.6.3.2 Remote Resume Request

The Remote Resume Request message shall be constructed as indicated in table 6-23.

Table 6-23: Remote Resume Request Message

Field	Size (bits)	Values	Comments
Reserved for future use	1	set to '0'	
Length of entity ID	3		Number of octets in source entity ID less one; i.e., '0' means that source entity ID is one octet.
Reserved for future use	1	set to '0'	
Length of Transaction sequence number	3		Number of octets in sequence number less one; i.e., '0' means that sequence number is one octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction that is to be resumed.
Transaction sequence number	variable		Uniquely identifies the transaction, among all transactions originated by this entity.

6.6.4 RESPONDING TO A REMOTE RESUME REQUEST

6.6.4.1 General

Upon receipt of a Remote Resume Request message, the remote CFDP user shall assure the resumption of the indicated transaction if possible; the CFDP `Resume.request` primitive will nominally be used for this purpose. It shall then use the CFDP `Put.request` primitive to request delivery of a file delivery unit to the entity which sent the original Resume Request message. The metadata of this file delivery unit shall contain one each of the Reserved CFDP Messages of message types Remote Resume Response (defined in 6.6.4.2) and Originating Transaction ID.

NOTE – See 6.5.4.1.2 for suspension order queue management requirements.

6.6.4.2 Remote Resume Response

The Remote Resume Response message shall be constructed as indicated in table 6-24.

Table 6-24: Remote Resume Response Message

Field	Size (bits)	Values	Comments
Suspension indicator	1	'0' — not suspended. '1' — suspended.	Because multiple motivations for suspending a transaction may be concurrently valid, the successful processing of a Remote Resume Request may not actually change the suspension status of the affected transaction.
Transaction status	2	Same values as defined for transaction status in the ACK PDU (see table 5-8).	
Spare	5	All 'zeroes'.	
Spare	1	'0'	This field and the following five fields identify the transaction whose resumption was requested; they are taken from the resume request.
Length of Entity ID	3		
Spare	1	'0'	
Length of Sequence Number	3		
Source Entity ID	Variable		
Transaction Sequence Number	Variable		

7 CFDP SERVICE CLASSES

NOTE – This section describes the minimum functionality of the CFDP protocol required to realize interoperable implementations of certain identified application services. The classes are, as yet, neither definitive nor complete. For each class, an event diagram is included to illustrate the sequence of events associated with the use of that protocol class.

7.1 DEFINED CLASSES

The defined CFDP protocol classes are:

a) Class 1—Unreliable Transfer;

b) Class 2—Reliable Transfer;

c) Class 3—Reliable Transfer by Proxy.

NOTES

1 State Transition Diagrams and State Tables for each of the Service Classes are provided in section 8.

2 The following subsections describe the functions and procedures, in sections 4 and 6 of this document, that are applicable to each Class. In order to implement the Class, it is only necessary to make reference to the General Characteristics overview of 2.3, the subsections specified in the Class procedures, and any PDU descriptions (section 5) referenced in the procedures.

7.2 FUNCTIONS OF CLASS 1—UNRELIABLE TRANSFER

NOTE – Class 1 provides for the unreliable delivery of bounded or unbounded data files from the source to the destination.

7.2.1 PROCEDURES FOR CLASS 1—SOURCE

For Class 1 functionality, the source CFDP entity shall use the procedures specified in table 7-1.

Table 7-1: Class 1 Source Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Transmitting Entity	4.1.1.1	
Put Procedures	4.1.2	
Transaction Start Notification Procedure	4.1.3	
PDU Forwarding Procedures	4.1.4	Applicable to Metadata, File Data and EOF PDUs only.
Copy File Procedures at the Sending Entity	4.1.5.1.1	
Unacknowledged Mode Procedures at the Sending Entity	4.1.5.3.1	
Fault Handling Procedures	4.1.7	Issuance of Notice of Cancellation is basic interoperable action.
Notice of Cancellation Procedures at the Sending Entity	4.1.10.2.2	Positive acknowledgement not required.

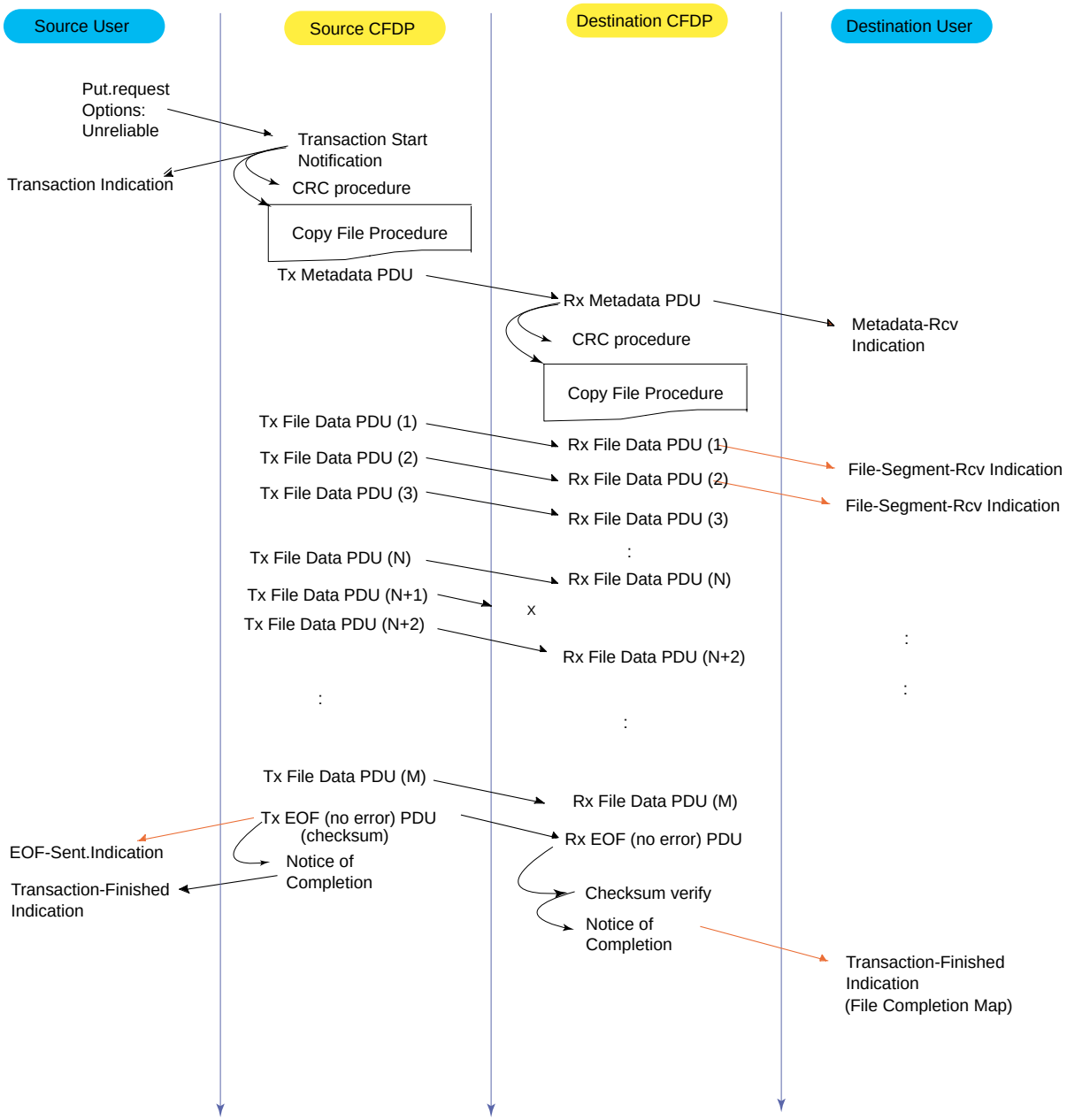
7.2.2 PROCEDURES FOR CLASS 1—DESTINATION

For Class 1 functionality, the destination CFDP entity shall use the procedures specified in table 7-2.

Table 7-2: Class 1 Destination Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Receiving Entity	4.1.1.2	
Copy File Procedures at the Receiving Entity	4.1.5.1.2	
Unacknowledged Mode Procedures at the Receiving Entity	4.1.5.3.2	
Cancel Response Procedures at the Receiving Entity	4.1.5.6.1	Positive acknowledgement not required.
Fault Handling Procedures	4.1.7	Abandonment of transaction is basic interoperable action.
Notice of Completion Procedures at the Receiving Entity	4.1.10.1.2	

7.2.3 EVENT DIAGRAM FOR CLASS 1



7.3 FUNCTIONS OF CLASS 2—RELIABLE TRANSFER

NOTE – Class 2 provides for the reliable delivery of bounded or unbounded data files from the source to the destination.

7.3.1 PROCEDURES FOR CLASS 2—SOURCE

For Class 2 functionality, the source CFDP entity shall use the procedures specified in table 7-3.

Table 7-3: Class 2 Source Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Transmitting Entity	4.1.1.1	
CRC Procedures at the PDU Receiving Entity	4.1.1.2	
Put Procedures	4.1.2	
Transaction Start Notification Procedure	4.1.3	
PDU Forwarding Procedures	4.1.4	
Copy File Procedures at the Sending Entity	4.1.5.1.1	
Acknowledged Mode Procedures at the Sending Entity	4.1.5.4.1	
Cancel Response Procedures at the Sending Entity	4.1.5.6.2	
Resume Procedures	4.1.5.7	Only procedures applicable to the sending entity.
Positive Acknowledgement Procedures at PDU Sending End	4.1.6.1	
Positive Acknowledgement Procedures at PDU Receiving End	4.1.6.2	
Fault Handling Procedures	4.1.7	Only issuance of Notice of Cancellation is required.
Internal Procedures	4.1.10	Only procedures applicable to the sending entity.

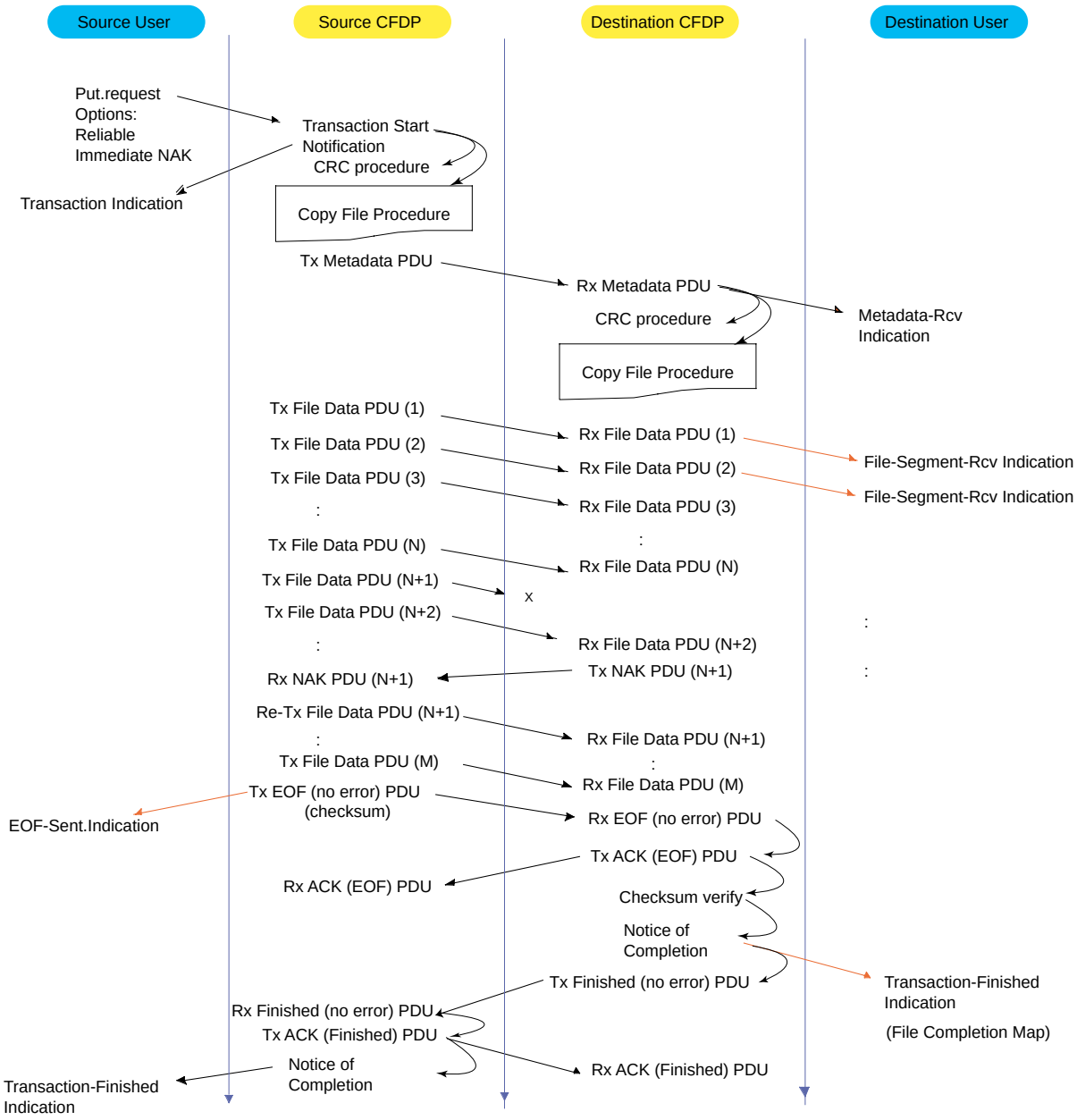
7.3.2 PROCEDURES FOR CLASS 2—DESTINATION

For Class 2 functionality, the destination CFDP entity shall use the procedures specified in table 7-4.

Table 7-4: Class 2 Destination Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Transmitting Entity	4.1.1.1	
CRC Procedures at the PDU Receiving Entity	4.1.1.2	
PDU Forwarding Procedures	4.1.4	
Copy File Procedures at the Receiving Entity	4.1.5.1.2	
Acknowledged Mode Procedures at the Receiving Entity	4.1.5.4.2	
Incremental Lost Segment Detection Procedures at the Receiving Entity	4.1.5.4.3	Deferred Mode required only for interoperability.
Cancel Response Procedures at the Receiving Entity	4.1.5.6.1	Positive acknowledgement required. Incomplete data forwarded.
Resume Procedures	4.1.5.7	Only procedures applicable to the receiving entity.
Positive Acknowledgement Procedures at PDU Sending End	4.1.6.1	
Positive Acknowledgement Procedures at PDU Receiving End	4.1.6.2	
Fault Handling Procedures	4.1.7	Only issuance of Notice of Cancellation is required.
Internal Procedures	4.1.10	Only procedures applicable to the receiving entity.

7.3.3 EVENT DIAGRAM FOR CLASS 2



7.4 FUNCTIONS OF CLASS 3—RELIABLE TRANSFER BY PROXY

NOTE – Class 3 provides for the reliable delivery of bounded or unbounded data files from a source (respondent) to the destination (beneficiary) under the initiation of a third entity (originator).

7.4.1 PROCEDURES FOR CLASS 3—BENEFICIARY

For Class 3 functionality, the beneficiary CFDP entity, i.e., the entity that receives the file transferred by proxy, shall use the procedures specified in table 7-5.

Table 7-5: Class 3 Beneficiary Procedures

Procedure	Subsection	Notes
Procedures for Class 2—Destination	7.3.2	

7.4.2 PROCEDURES FOR CLASS 3—RESPONDENT (PROXY)

For Class 3 functionality, the respondent (i.e., proxy) CFDP entity, i.e., the entity that transfers the file, shall use the procedures specified in table 7-6.

Table 7-6: Class 3 Respondent Procedures

Procedure	Subsection	Notes
Responding to a Remote Put Order	6.2.4	
Reporting on the Results of a Remote Put Order	6.2.5	
Procedures for Class 2—Source	7.3.1	
Procedures for Class 2—Destination	7.3.2	

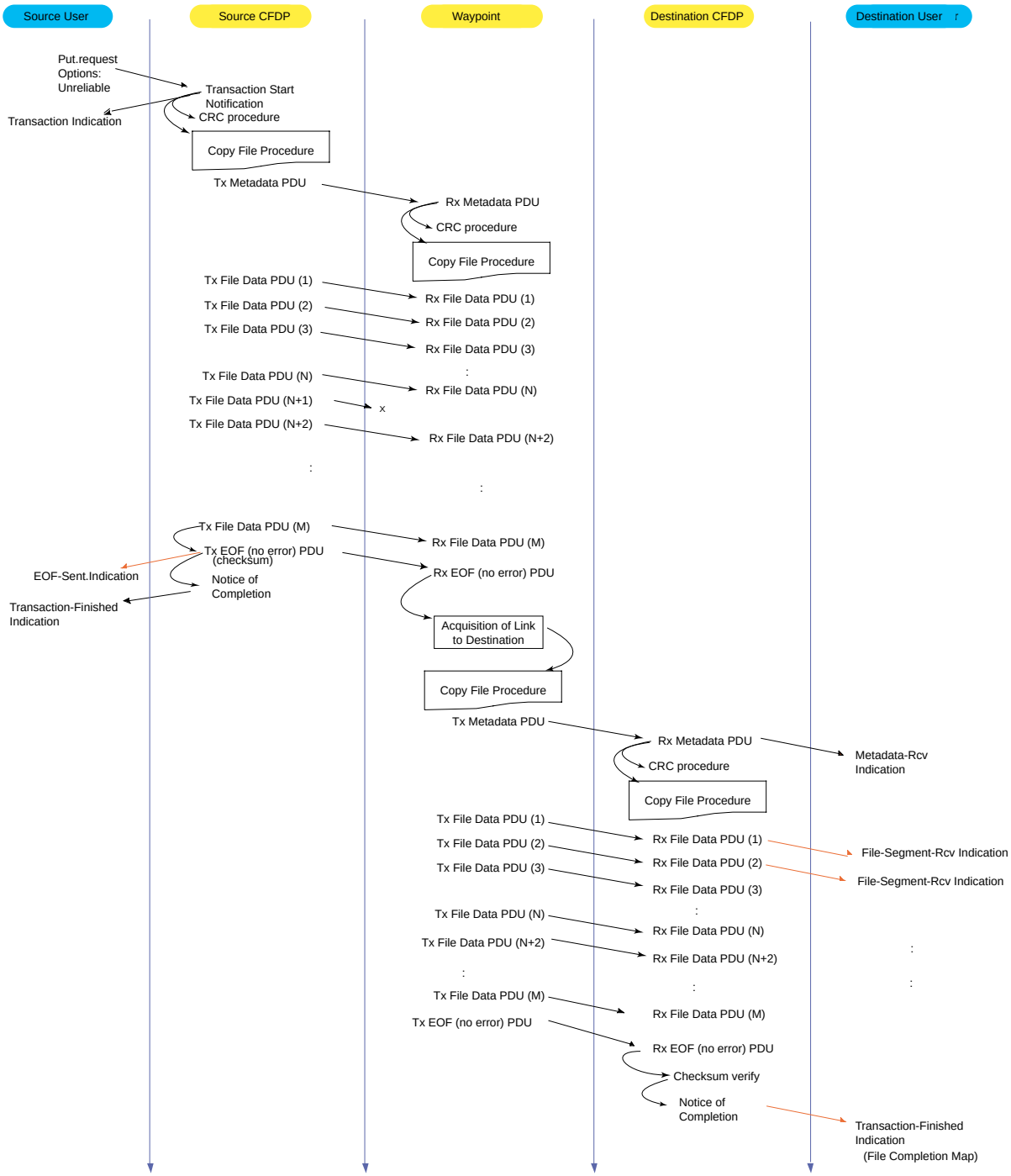
7.4.3 PROCEDURES FOR CLASS 3—ORIGINATOR

For Class 3 functionality, the originating CFDP entity, i.e., the entity that requests the file transfer by proxy, shall use the procedures specified in table 7-7.

Table 7-7: Class 3 Originator Procedures

Procedure	Subsection	Notes
Initiating a Proxy Operation	6.2.3	
Procedures for Class 2—Source	7.3.1	
Procedures for Class 2—Destination	7.3.2	

7.4.4 EVENT DIAGRAM FOR CLASS 3



8 CFDP STATE DIAGRAMS AND TABLES

8.1 GENERAL

This section contains two sets of CFDP state tables/diagrams. The first set (8.2) is intended to provide a technically definitive description of the protocol. The second set (8.3) is intended to be supplementary in nature as a more easily understood presentation.

8.2 DEFINING CFDP STATE DIAGRAMS AND TABLES

8.2.1 INTRODUCTION

8.2.1.1 Overview

State Tables define the actions to be taken within an existing transaction in response to both external events (e.g., receipt of PDUs and User Requests) and internal events (e.g., the ACK-timer expires). The State Table to be used for a particular transaction depends upon the CFDP entity's role in that transaction. Service Classes 1 and 2 are covered as shown in table 8-1.

Table 8-1: State Tables by Service Class

Class	Source	Destination
1	Figure 8-1: Unacknowledged Mode, Sender	Figure 8-2: Unacknowledged Mode, Receiver
2	Figure 8-3: Acknowledged Mode, Sender	Figure 8-4: Acknowledged Mode, Receiver

8.2.1.2 State Machines

When State Table logic is used for a particular transaction, it is referred to as a *State Machine.* There is only one definition of each State Table, but there can be many State Machines running (each State Machine is assigned to one transaction).

8.2.1.3 Kernel

8.2.1.3.1 The State Tables assume that some other part of the CFDP implementation:

- starts up all required State Machines;
- establishes the UT-layer connection(s);

- receives all incoming PDUs via the UT layer, and delivers them to the appropriate State Machine(s) by triggering the appropriate event(s);
- receives all User Requests and delivers them to the appropriate State Machine(s) by triggering the appropriate event(s).

8.2.1.3.2 The Recommendation does not specify how these needs are met, but the concept is referred to as the kernel. Sample kernel logic is provided in reference [A4].

8.2.2 STATE DIAGRAMS

The State-Transition Diagrams summarize the State Tables.

General notes:

- Heavy lines are used to show the 'nominal' path through the state diagram for a transaction involving file transfer. The nominal path assumes that no transmission errors occur (i.e., no messages missed, no bit errors, no sequencing errors, etc.).
- Text indicates what causes each state change. If the text is in the form of a 'fraction', then the 'numerator' indicates what caused the state change, and the 'denominator' indicates an action to be taken.
- All PDU transmissions are shown. The receipt of PDUs is shown except in cases where they are to be ignored.
- Abbreviations for PDUs: MD=Metadata, FD=File data.
- All the real work accomplished by a transaction (i.e., delivering User Messages, a file, and/or Filestore Requests) is contained within 2 actions:
  - 'PM' means 'process Metadata' (this includes delivering any User Messages contained in the Metadata PDU).
  - 'Process Data' means 'validate/accept the received file (if any) and execute Filestore Directives (if any)'.
- Some aspects of the protocol are not shown (e.g., Cancel, Suspend/Resume, Fault Handling, use of the Inactivity Timer, optional PDUs such as Keep Alive, etc.).

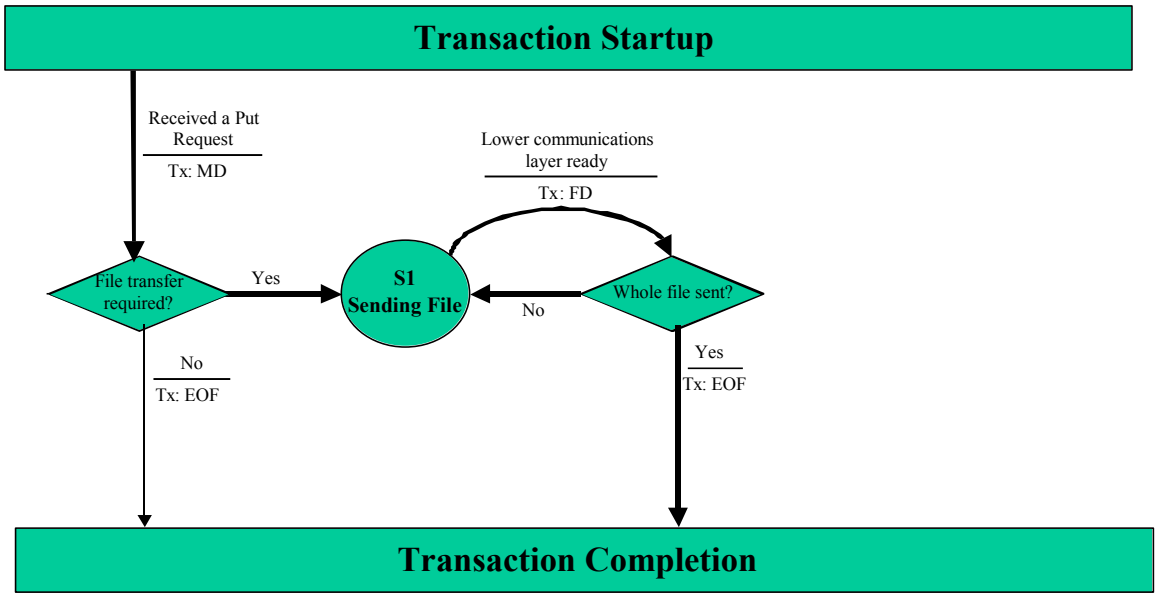


Figure 8-1: Unacknowledged Mode, Sender

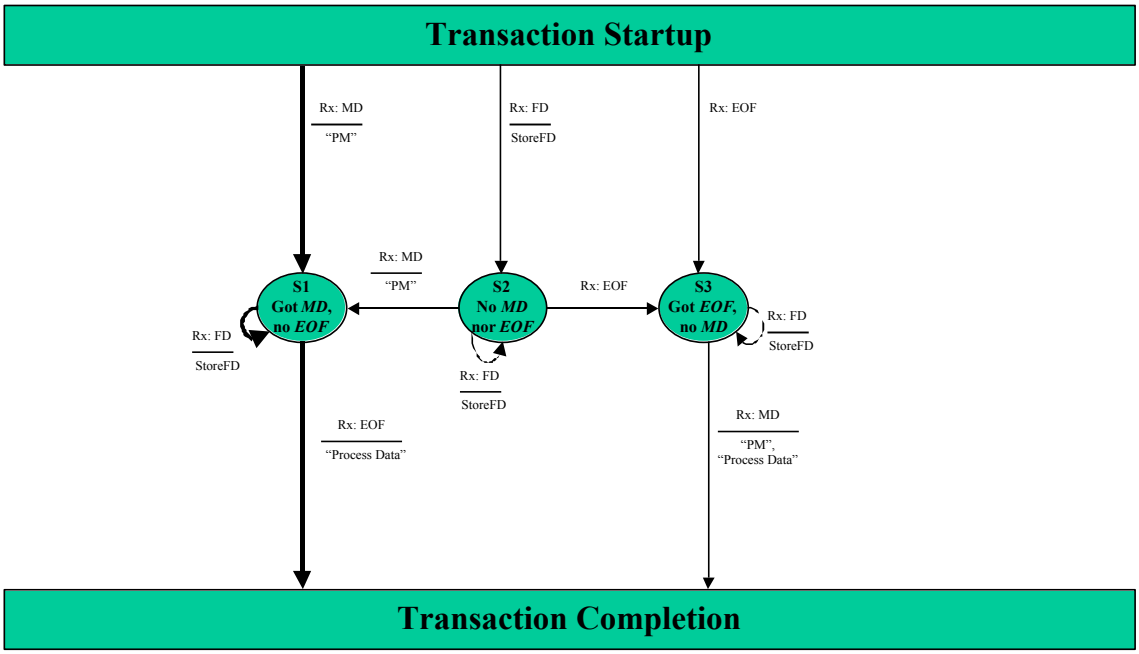


Figure 8-2: Unacknowledged Mode, Receiver

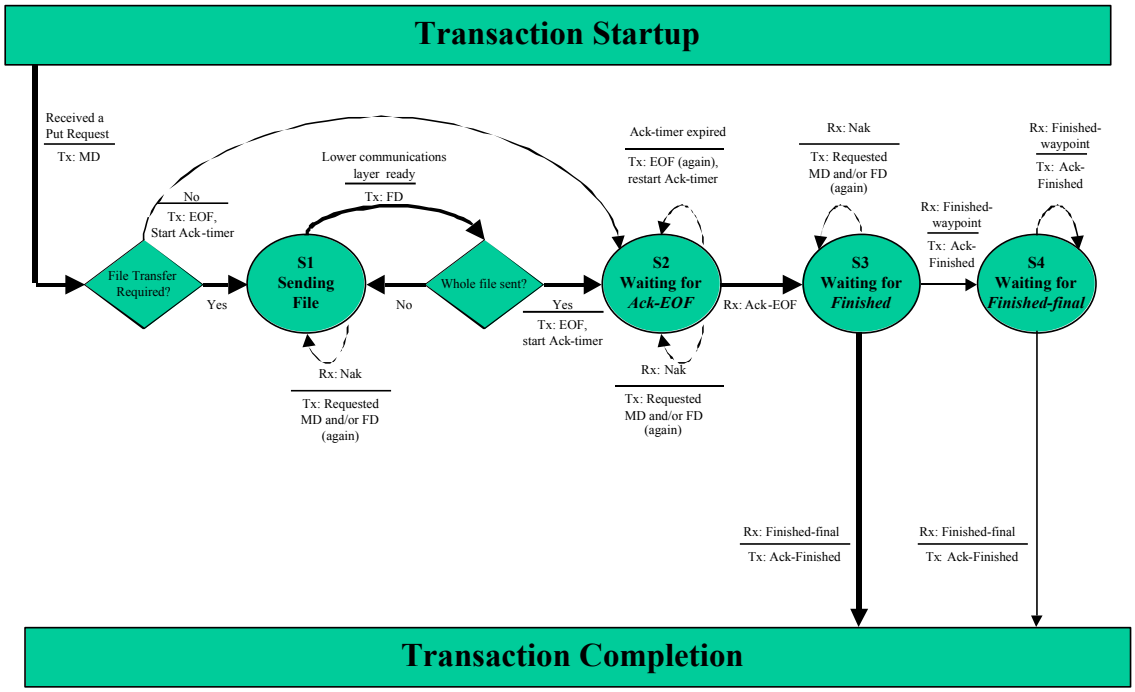
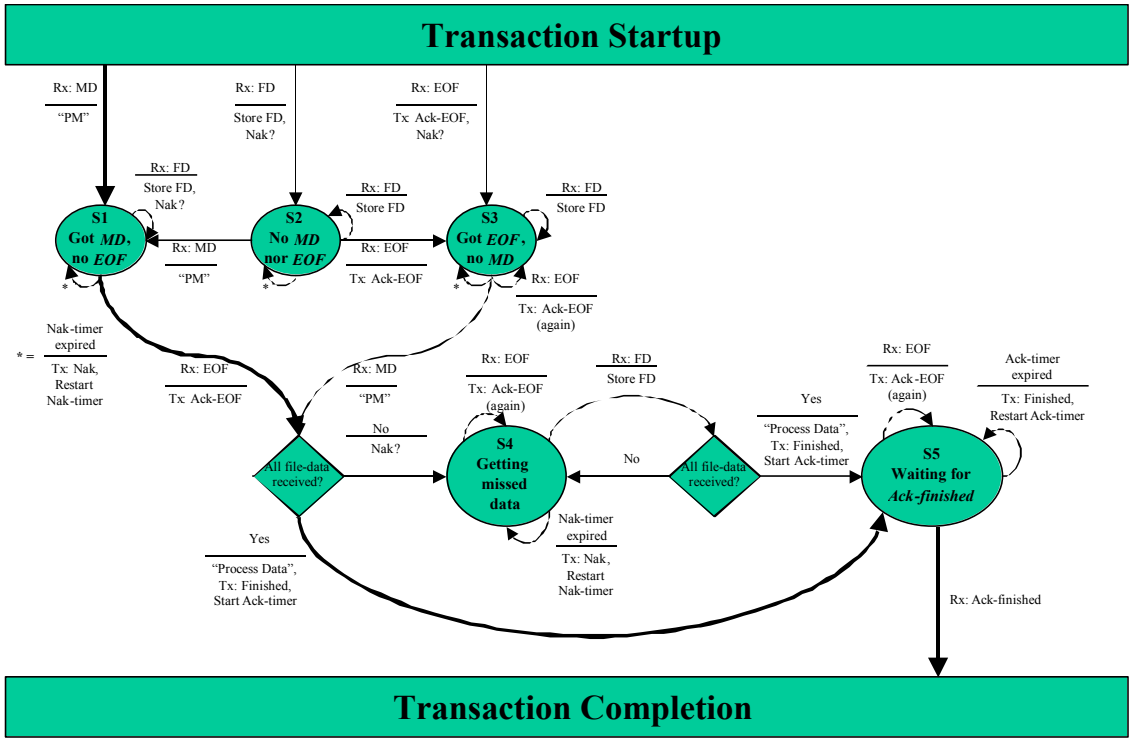


Figure 8-3: Acknowledged Mode, Sender



NOTE – 'NAK?' = A NAK PDU is sent if Metadata and/or File-data is missing AND it is appropriate to do so AT THIS TIME in the current NAK-mode. If a NAK PDU is sent, then the NAK-timer is started. The NAK-timer ensures periodic feedback to the sender until all data has been received.

Figure 8-4: Acknowledged Mode, Receiver

8.2.3 STATE TABLES

8.2.3.1 General

8.2.3.1.1 A set of states is defined as appropriate for each State Table (i.e., state S2 in one table may have a different meaning than state S2 in another table).

8.2.3.1.2 A single set of events is defined (i.e., event E12 always means the same thing).

8.2.3.1.3 General comments applying to all of the State Tables:

- All State Machines begin in State S0.
- For each Event + State combination, the required actions and state changes are specified (except for state changes possible via the 'Fault' action).
- Whenever a state change occurs, there is to be an implicit triggering of event E0, allowing actions to be performed upon entering the new state. When a state change occurs, Event E0 must be triggered and handled before moving on to the next event (e.g., if events are queued, then E0 always goes to the front of the queue).
- For those events omitted from a State Table, no action is required.

8.2.3.2 State Table Terminology

- 'N/A'—No action required.
- 'Send …'—Generate the specified PDU, queue it for transmission to the partner CFDP entity, and store a copy if retransmission may be necessary.
- 'Issue …'—Generate the specified Indication and send it to the User.
- 'Trigger …'—Deliver the specified event (back to itself, unless otherwise specified). The implementer can choose how to deliver the event, but the triggered event must not begin until the current event finishes.

8.2.3.3 Key Actions

The emphasis in the State Tables is on responding properly to the partner CFDP entity (to ensure reliable delivery of PDUs). Detailed specifications for such things as 'what do I do with this Metadata and File-data now that it has been delivered to me?' can be found in the remainder of section 4. A summary of the key actions:

'Process Metadata'—Includes setting transaction Fault-Handling according to any Fault-Handler-Override TLVs included in the Metadata, delivering any User Messages to the User (if we are the final destination), and storing any Filestore Requests included in the Metadata for possible later execution.

‘Process Data’—Includes validating and possibly storing any file which has been transferred, executing any Filestore Requests (unless file validation failed), and possibly generating Filestore Status information (regarding the Filestore Requests which were executed).

‘Store new file-data’—When a section of the file is received for the first time, it is stored. If the same section of data is received again, it is discarded.

‘Is UT layer ready’—Implementation-dependent. The intentions are to meter a file’s data across the link layer and to get ‘protocol control messages’ across the link quickly (e.g., an *EOF-Cancel* PDU should not wait in a transmission buffer behind 2 megabytes worth of *File-data* PDUs).

‘Fault’—Determined by what fault condition was declared and the Fault-Handler settings. The MIB contains default Fault-Handler settings; these may have been overridden by Fault-Handler-Override TLVs in the Metadata PDU.

‘Initialize’—Implementation-dependent.

‘Shutdown’—Implementation-dependent, but must include issuing a Transaction-`Finished.indication` (if possible) and the stopping of any timers. Note that it may be necessary to let the kernel know that this State Machine is shutting down.

‘Respond to inactivity timeout’—Implementation-dependent, but must include issuing a `Fault.indication`.

‘Dispose of the file’—Implementation-dependent; could result in discarding the file.

Table 8-2: Unacknowledged Mode State Table, Sender

State: Event:	S0 Transaction Startup	S1 Transferring File
E0 Entered this State (an implicit event whenever a state change occurs)	Initialize.	Trigger E1

E1
Please send some file-data
	N/A	If (Is UT layer ready)
		Send one File-data PDU,
		If (has entire file been sent)
		Send EOF,
		Shutdown
		Else
		Trigger E1
		Else
		Sleep for a short time
		Trigger E1

E11
Rx: Put Request
(This is the first event received)
	If (Is the Put Request valid)
	Issue Transaction-Started,
	Send Metadata,
	If (Is file transfer needed)
	Prepare to send the file,
	State=S1
	Else
	Send EOF,
	Shutdown
	Else
	Shutdown
		N/A

E29 Rx: Cancel Request	N/A	Send EOF-cancel, Shutdown
E36 Rx: Report Request	N/A	Issue report

Table 8-3: Unacknowledged Mode State Table, Receiver

State: Event:	S0 Transaction Startup	S1 Got MD, no EOF	S2 No MD nor EOF	S3 Got EOF, no MD
E0 Entered this State (an implicit event whenever a state change occurs)	Initialize.	N/A	N/A	N/A

E12
Rx: Metadata PDU
(This is normally the first event received)
	If (Is Metadata PDU valid)
	Issue Metadata-Recv,
	Process Metadata,
	State=S1
	Else
	Fault
		N/A	If (Is Metadata PDU valid)
			Issue Metadata-Recv,
			Process Metadata,
			State=S1
			Else
			Fault
				If (Is Metadata PDU valid)
				Issue Metadata-Recv,
				Process Metadata,
				Process Data,
				Shutdown
				Else
				Fault

E13 Rx: File-Data PDU	Issue File-Seg-Recv, Store new file-data, State=S2	Issue File-Seg-Recv, Store new file-data	Issue File-Seg-Recv, Store new file-data	Issue File-Seg-Recv, Store new file-data
E14 Rx: EOF PDU	State=S3	Process Data, Shutdown	State=S3	N/A
E23 Inactivity-timeout	N/A	Respond to inactivity timeout	Respond to inactivity timeout	Respond to inactivity timeout
E26 Rx: EOF-Cancel PDU	N/A	Dispose of the file, Shutdown	Dispose of the file, Shutdown	Dispose of the file, Shutdown
E36 Rx: Report Request	N/A	Issue Report	Issue Report	Issue Report

Table 8-4: Acknowledged Mode State Table, Sender

State: Event:	S0 Transaction Startup	S1 Sending File	S2 Waiting for ACK-EOF	S3 Waiting for Finished	S4 Waiting for Finished-final	S5 Canceling
E0 Entered this state (implicit)	Initialize.	Trigger E1	Send EOF, Set Attempts=1, Start ACK-timer	N/A	N/A	Send EOF-Cancel, Set Attempts=1, Start ACK-timer

E1
Is UT layer ready?
	N/A	If (Is UT layer ready)
		Send one File-data PDU,
		If (Has entire file been sent)
		State=S2
		Else
		Trigger E1
		Else
		Sleep for a short time,
		Trigger E1
			N/A	N/A	N/A	N/A

E11
Rx: Put Request
(This is the first event received)
	If (Is the Put Request valid)
	Issue Transaction,
	Send Metadata,
	If (Is file transfer needed)
	Prepare to send the file,
	State=S1
	Else
	State=S2
	Else
	Shutdown
		N/A	N/A	N/A	N/A	N/A

E15 Rx: ACK-EOF PDU	N/A	N/A	Stop ACK-timer State=S3	N/A	N/A	N/A
E16 Rx: Finished-final PDU	N/A	N/A	N/A	Send ACK-Finished, Shutdown	Send ACK-Finished, Shutdown	N/A
E17 Rx: ACK-Finished PDU	N/A	N/A	N/A	N/A	N/A	Shutdown

Table 8-4: Acknowledged Mode State Table, Sender (continued)

State: Event:	S0 Transaction Startup	S1 Sending File	S2 Waiting for ACK-EOF	S3 Waiting for Finished	S4 Waiting for Finished-final	S5 Canceling

E18
Rx: NAK PDU
	N/A	If (Does NAK include Metadata)
		If (Attempts < Max_attempts)
		Resend Metadata, Increment Attempts
		Else
		Fault
		If (Does NAK include File-data)
		Resend file-data
			If (Does NAK include Metadata)
			If (Attempts < Max_attempts)
			Resend Metadata, Increment Attempts
			Else
			Fault
			If (Does NAK include File-data)
			If (Is file transfer needed)
			Resend file-data
				If (Does NAK include Metadata)
				If (Attempts < Max_attempts)
				Resend Metadata, Increment Attempts
				Else
				Fault
				If (Does NAK include File-data)
				If (Is file transfer needed)
				Resend file-data
					N/A	N/A

E21
ACK-timeout
	N/A	N/A	If (attempts < Max_Attempts)
			Resend EOF, Increment Attempts, Start ACK-timer
			Else
			Fault
				N/A	N/A	If (attempts < Max_Attempts)
						Resend EOF-Cancel, Increment Attempts, Start ACK-timer
						Else
						Shutdown

E23 Inactivity Timeout	N/A	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	N/A
E28 Rx: Finished-Cancel PDU	N/A	Send ACK-Finished,, Shutdown	Send ACK-Finished,, Shutdown	Send ACK-Finished,, Shutdown	N/A	Send ACK-Finished,, Shutdown
E29 Rx: Cancel Request	N/A	State=S5	N/A	N/A	N/A	N/A
E31 Rx: Suspend Request	N/A	TBD	TBD	TBD	TBD	N/A
E34 Rx: Resume Request	N/A	TBD	TBD	TBD	TBD	N/A
E36 Rx: Report Request	N/A	Issue report	Issue report	Issue report	Issue report	Issue report
E41 Please send Prompt-NAK	N/A	Send Prompt-NAK	Send Prompt-NAK	Send Prompt-NAK	N/A	N/A

Table 8-4: Acknowledged Mode State Table, Sender (continued)

State: Event:	S0 Transaction Startup	S1 Sending File	S2 Waiting for ACK-EOF	S3 Waiting for Finished	S4 Waiting for Finished-final	S5 Canceling
E44 Please send Prompt-Keep Alive	N/A	Send Prompt-Keep Alive	Send Prompt-Keep Alive	Send Prompt-Keep Alive	N/A	N/A
E46 Rx: Keep Alive PDU	N/A	If (!Is Keep Alive offset valid) Fault	If (!Is Keep Alive offset valid) Fault	N/A	N/A	N/A
E51 Rx: Finished PDU	N/A	N/A	N/A	Send ACK-Finished, Issue Transfer-Consigned, State=S4	Send ACK-Finished	N/A

Table 8-5: Acknowledged Mode State Table, Receiver

State: Event:	S0 Transaction Startup	S1 Got MD, no EOF	S2 No MD nor EOF	S3 Got EOF, no MD	S4 Getting missed data	S5 Waiting for ACK-Finished	S6 Canceling
E0 Entered this State (implicit)	Initialize.	N/A	N/A	N/A	N/A	Process data, Send Finished, Start ACK-timer	Send Finished-Cancel, Set Attempts=1, Start ACK-timer
E2 Please send NAK	N/A	Send NAK, Increment NAK-count, Start NAK-timer	Send NAK, Increment NAK-count, Start NAK-timer	Send NAK, Increment NAK-count, Start NAK-timer	Send NAK, Increment NAK-count, Start NAK-timer	N/A	N/A

E3
All File-data received?
	N/A	If (Has all file-data been
		received)
		State=S5
		Else
		if (NAK_mode==Deferred)
		Trigger E2
		State=S4
			N/A	If (Has all file-data been
				received)
				State=S5
				Else
				if (NAK_mode==Deferred)
				Trigger E2
				State=S4
					If (Has all file-data been
					received)
					State=S5
						N/A	N/A

E12
Rx: Metadata PDU
(This is normally the first event received)
	If (Is Metadata PDU valid)
	Issue Metadata-Recv,
	Process Metadata,
	State=S1
	Else
	Fault
		N/A	If (Is Metadata PDU valid)
			Issue Metadata-Recv,
			Process Metadata,
			State=S1
			Else
			Fault
				If (Is Metadata PDU valid)
				Issue Metadata-Recv,
				Process Metadata,
				Trigger E3
				Else
				Fault
					N/A	N/A	N/A

Table 8-5: Acknowledged Mode State Table, Receiver (continued)

State: Event:	S0 Transaction Startup	S1 Got MD, no EOF	S2 No MD nor EOF	S3 Got EOF, no MD	S4 Getting missed data	S5 Waiting for ACK-Finished	S6 Canceling
E13 Rx: File-Data PDU	Issue File-Seg-Recv, Store new file-data, If (NAK_mode == Immediate) Trigger E2 State=S2	Issue File-Seg-Recv, Store new file-data, If ((Is there a new file gap) & (NAK_mode == Immediate)) Trigger E2	Issue File-Seg-Recv, Store new file-data, If ((Is there a new file gap) & (NAK_mode == Immediate)) Trigger E2	Issue File-Seg-Recv, Store new file-data, If ((Is there a new file gap) & (NAK_mode == Immediate)) Trigger E2	Issue File-Seg-Recv, Store new file-data, Trigger E3	N/A	N/A
E14 Rx: EOF PDU	Send ACK-EOF, Trigger E2, State=S3	Send ACK-EOF, Trigger E3	Send ACK-EOF, State=S3	Resend ACK-EOF	Resend ACK-EOF	Resend ACK-EOF	N/A
E17 Rx: ACK-Finished PDU	N/A	N/A	N/A	N/A	N/A	Shutdown	Shutdown

E21
ACK-timeout (i.e., Partner has not responded)
	N/A	N/A	N/A	N/A	N/A	If (attempts < Max_Attempts)
						Resend Finished,
						Increment attempts,
						Start ACK-timer
						Else
						Fault
							If (attempts < Max_Attempts)
							Resend Finished-Cancel
							Increment attempts,
							Start ACK-timer
							Else
							Shutdown

E22
NAK-timeout (i.e., Periodic feedback to partner)
	N/A	If (NAK-count < Max_NAKs)
		Trigger E2
		Else
		Fault
			If (NAK-count < Max_NAKs)
			Trigger E2
			Else
			Fault
				If (NAK-count < Max_NAKs)
				Trigger E2
				Else
				Fault
					N/A	N/A	N/A

Table 8-5: Acknowledged Mode State Table, Receiver (continued)

State: Event:	S0 Transaction Startup	S1 Got MD, no EOF	S2 No MD nor EOF	S3 Got EOF, no MD	S4 Getting missed data	S5 Waiting for ACK-Finished	S6 Canceling
E23 Inactivity Timeout	N/A	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	N/A
E26 Rx: EOF-Cancel PDU	N/A	Send ACK-EOF-Cancel, Shutdown	Send ACK-EOF-Cancel, Shutdown	Send ACK-EOF-Cancel, Shutdown	Send ACK-EOF-Cancel, Shutdown	Send ACK-EOF-Cancel, Shutdown	Send ACK-EOF-Cancel, Shutdown
E29 Rx: Cancel Request	N/A	State=S6	State=S6	State=S6	State=S6	N/A	N/A
E31 Rx: Suspend Request	N/A	TBD	TBD	TBD	TBD	TBD	N/A
E34 Rx: Resume Request	N/A	TBD	TBD	TBD	TBD	TBD	N/A
E36 Rx: Report Request	N/A	Issue report	Issue report	Issue report	Issue report	Issue report	Issue report
E42 Rx: Prompt-NAK PDU	N/A	If (NAK_mode==Prompted) Trigger E2	If (NAK_mode==Prompted) Trigger E2	If (NAK_mode==Prompted) Trigger E2	If (NAK_mode==Prompted) Trigger E2	N/A	N/A
E43 Asynchronous-NAK signal	N/A	If (NAK_mode==Asynch) Trigger E2	If (NAK_mode==Asynch) Trigger E2	If (NAK_mode==Asynch) Trigger E2	If (NAK_mode==Asynch) Trigger E2	N/A	N/A
E45 Rx: Prompt-Keep Alive PDU	N/A	Send Keep Alive	Send Keep Alive	Send Keep Alive	Send Keep Alive	Send Keep Alive	N/A

8.2.4 EVENTS

8.2.4.1 General

A single set of events is defined (i.e., event E12 is the same for all State Tables).

8.2.4.2 Implicit Event

The following event is implicit whenever a state change occurs; it enables actions to be performed upon entering a state. Resumption of a transaction is also a state change.

E0 – Entered this state.

8.2.4.3 Derived Events

The following events are derived from other events. For example, E3 might be derived from E13, assuming that the File-data PDU received via E13 also filled the last remaining file gap. They are generated and received by the same State Machine.

E1 – Please send some file-data. Senders use this event (internally) to send one File-data PDU at a time until the entire file is sent.

E2 – Please send a NAK. Receivers trigger this event whenever a NAK should be sent.

E3 – Both Metadata and Complete File received. Receivers trigger this event.

E4-E10 – Reserved for future expansion.

8.2.4.4 Transaction Startup

All transactions are started by the following event (receipt of a Put Request from the CFDP User):

E11 – Put Request.

8.2.4.5 PDUs

The following events are triggered by the receipt of the PDUs from the partner CFDP entity:

E12 – Metadata PDU.

E13 – File-data PDU.

E14 – EOF PDU (see ‘EOF-Cancel’ PDU).

E15 – ACK-EOF PDU (see ‘ACK-EOF-Cancel’ PDU).

E16 – Finished PDU from final destination (see ‘Finished-Cancel’ PDU).

E17 – ACK-Finished PDU.

E18 – NAK PDU.

E19-E20 – Reserved for future expansion.

8.2.4.6 Timeouts

The following events are triggered when internal timers expire. They are generated and received by the same State machine.

E21 – ACK-timeout.

E22 – NAK-timeout.

E23 – Inactivity-timeout.

E24-E25 – Reserved for future expansion.

8.2.4.7 Cancel/Suspend/Resume

E26 – EOF-Cancel PDU (EOF PDU indicating an aborted transaction).

E27 – ACK-EOF-Cancel PDU (ACK PDU indicating an ACK of an EOF-Cancel).

E28 – Finished-Cancel PDU (Finished PDU indicating an aborted transaction).

NOTE – A Finished-Cancel PDU is ACKed by an ACK-Finished PDU.

E29 – Cancel Request.

E31 – Suspend Request.

E34 – Resume Request.

E36 – Report Request.

E37-E40 – Reserved for future expansion.

8.2.4.8 Optional Events

The following events are triggered by optional features.

E41 – Please send a Prompt-NAK PDU.

E42 – Received a Prompt-NAK PDU.

E43 – Received an Asynchronous-NAK signal.

E44 – Please send a Prompt-Keep Alive PDU.

E45 – Received a Prompt-Keep Alive PDU.

E46 – Received a Keep Alive PDU.

E47-E50 – Reserved for future expansion.

8.3 A SDL/GR REPRESENTATION OF CFDP STATE DIAGRAMS

8.3.1 PURPOSE AND SCOPE

8.3.1.1 This subsection provides state diagrams of the Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol (CFDP) Entities using the ITU SDL graphical representation technique (see reference [A5]). These representations are not intended as replacements for the natural-language specifications provided in the other sections of this document, but as a pilot boat to navigate specifications for implementers.

8.3.1.2 These representations describe procedures for Class 1-Source, Class 1-Destination, Class 2-Source and Class 2-Destination entities with following MIB settings:

Default fault handlers—a Notice of Cancellation,

Immediate NAK mode enabled—*yes*,

Prompt NAK mode enabled—*no*,

Asynchronous NAK mode enabled—*no*,

CRCs required on transmission—*false*.

8.3.2 STATE DIAGRAM TERMINOLOGY

8.3.2.1 Interfaces

'UI' – Interface for CFDP SERVICE PRIMITIVES (see 3.5).

'UT' – Interface for CFDP PDUs (see section 5).

8.3.2.2 Variables

'condition' – a condition code (see table 5-5).

'attempt' – a retry counter for an ACK or NAK sequence (see 4.1.6.1, 4.1.5.4.2).

'limit' – an ACK or NAK timer expiration limit (see table 9-1).

'PDU-recved' – a variable which indicates whether the destination entity received any file data or metadata PDU after an NAK-timer started (see 4.1.5.4.2.2).

8.3.2.3 Timer Operations

'set' – preset and start (or restart) an ACK or NAK timer (see 4.1.5.4).

'reset' – reset and stop an ACK or NAK timer (see 4.1.10).

8.3.2.4 Decisions

'file structure' – whether or not the source entity did observe record structure to the source file, which the segmentation control service parameter has requested via `Put.request` (see 4.1.5.1.1.7).

'acceptable file' – whether or not the destination entity (filestore) could accept the file (see 4.1.5.1.2.3).

'checksum' – whether or not the calculated and received checksums are equal at the destination entity (see 4.1.5.1.2.4).

'file length' – whether or not the count of octets received is greater than the file size indicated in the EOF(No error) PDU at the destination entity (see 4.1.5.1.2.5).

'sent entire file' – whether or not the source entity has sent entire file (see 4.1.5.1.1.8).

'repeated FD' – whether or not the received file segments are already received at the destination entity (see 4.1.5.1.2.4).

‘received all’ – whether or not the destination entity deems to have received the metadata and all file segments (see 4.1.5.4.2.3).

‘requested all’ – whether or not the destination entity has already requested all lost segments by means of NAK(s) (see 4.1.5.4.2.2).

‘detected new gap’ – whether or not the destination entity detected new lost segment (see 4.1.5.4.3).

8.3.2.5 Procedures

‘release re-TX buffer’ – the source entity releases a retransmission buffer (see 4.1.10).

‘detect lost file data’ – the source entity recognizes lost file segments and/or metadata according to received NAK(s) (see 4.1.5.4.1).

‘save gap’ – the destination entity stores or updates offsets of lost segments for later (see 4.1.5.4.3).

‘process MD’ – the destination entity interprets metadata and saves information for later (see 4.1.5.1.2).

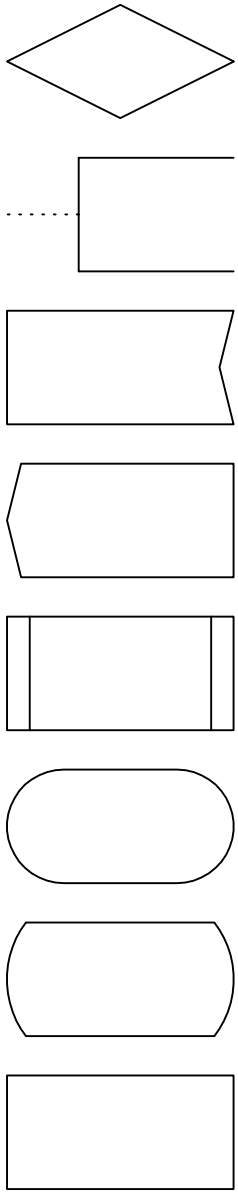
‘process FD’ – the destination entity interprets a file segment offset and stores the contents of file (see 4.1.5.1.2).

8.3.2.6 Abbreviations

‘TR-FIN’ – `Transaction-Finished.indication`.

8.3.3 GRAPHICAL SYMBOL CONVENTION

This subsection describes a summary of graphical symbols used in following state diagrams. Detail information about symbols are described in the SDL recommendations.



Decision symbol

Comment symbol

Input symbol

Output symbol(dotted: optional)

Procedure call symbol

Start symbol

State symbol

Task symbol

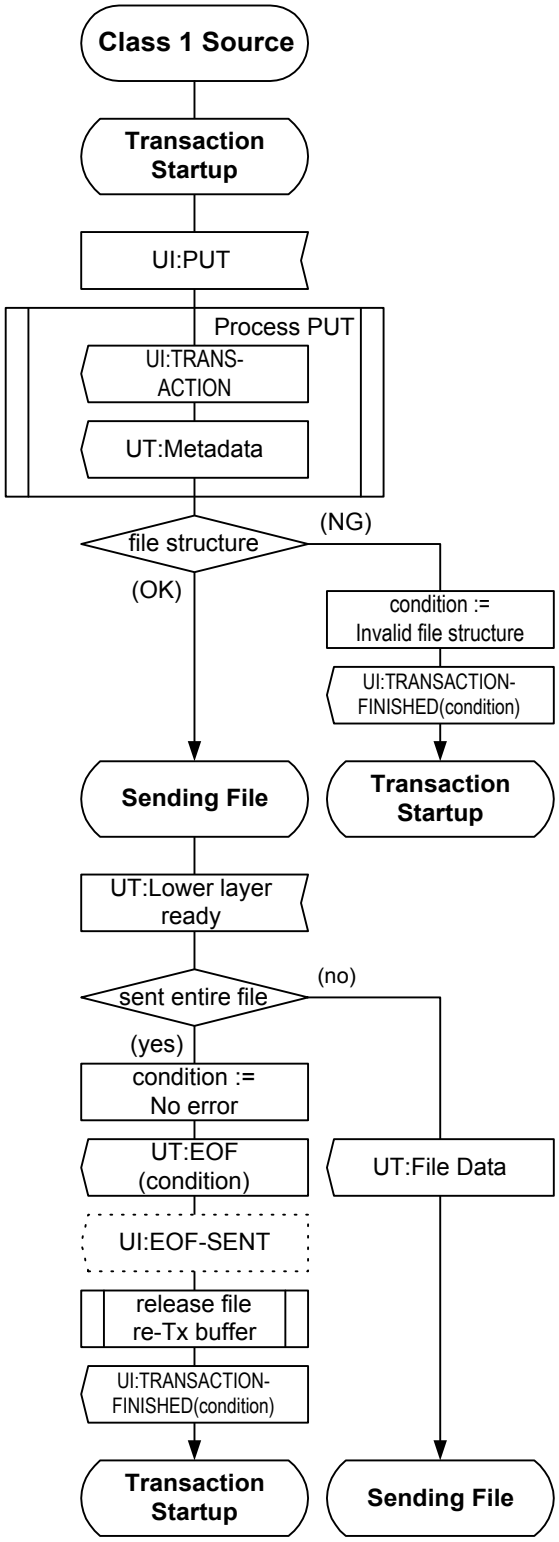


Figure 8-5: Class 1 Source State Diagram

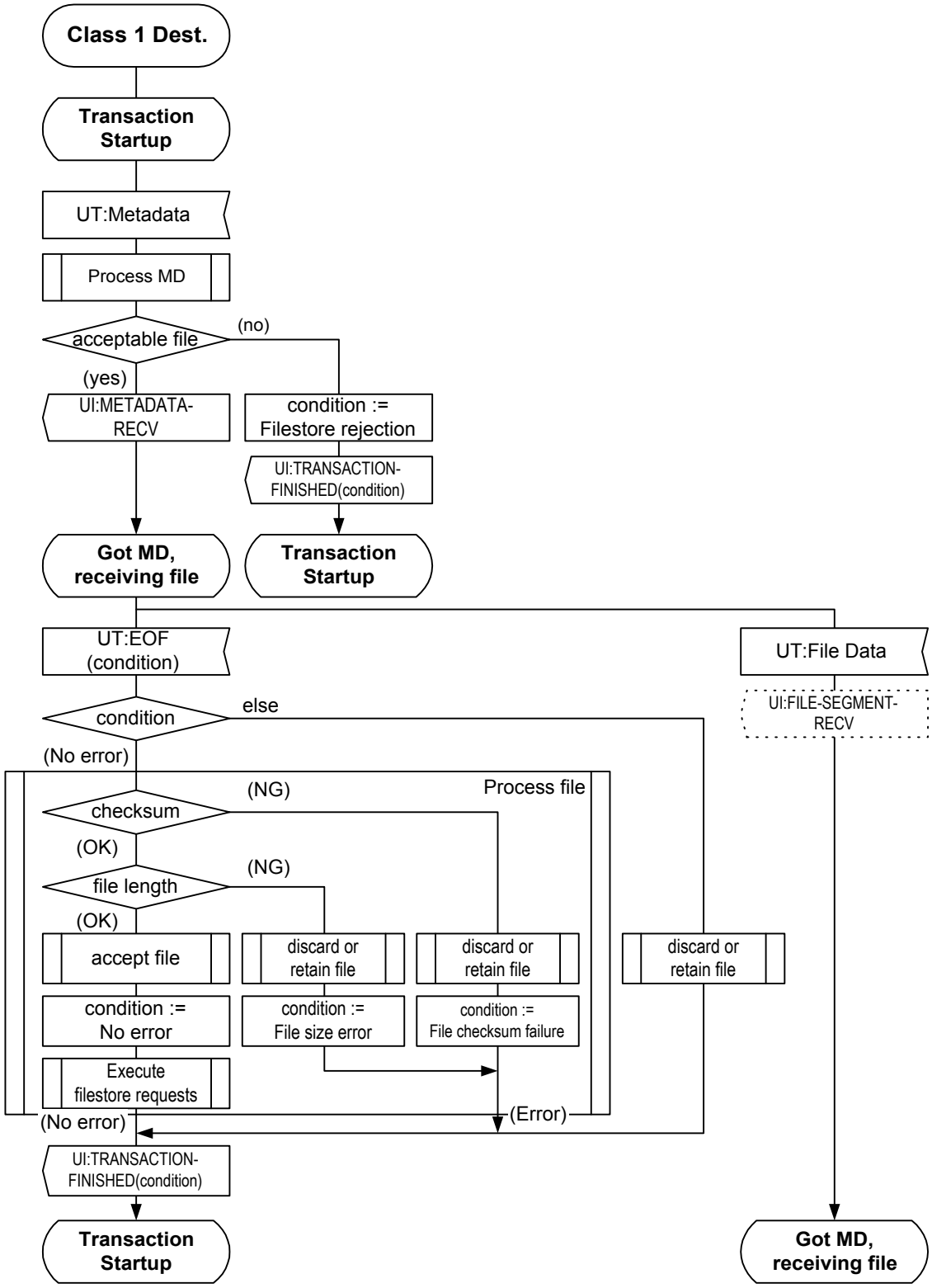


Figure 8-6: Class 1 Destination State Diagram

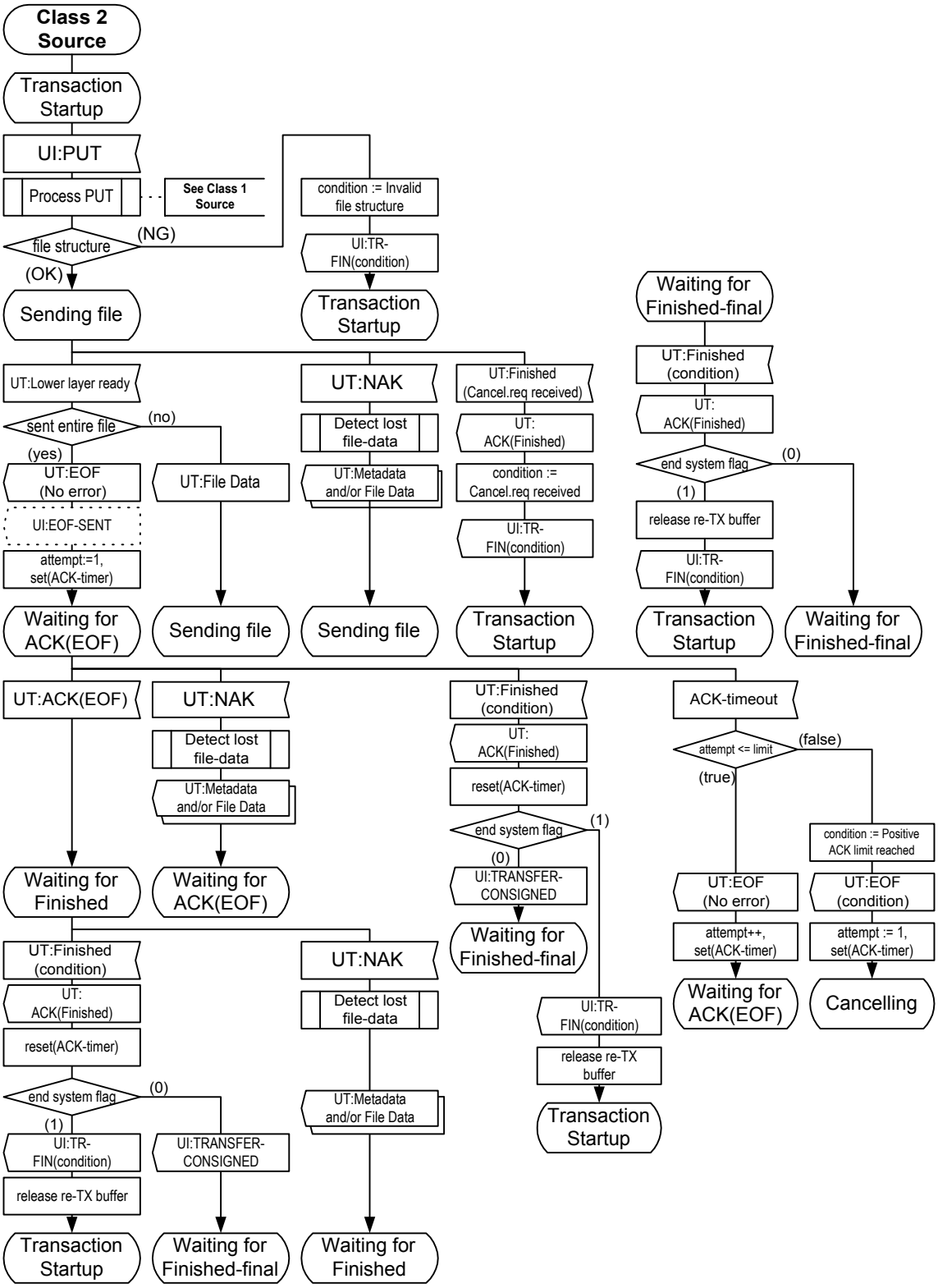


Figure 8-7: Class 2 Source State Diagram

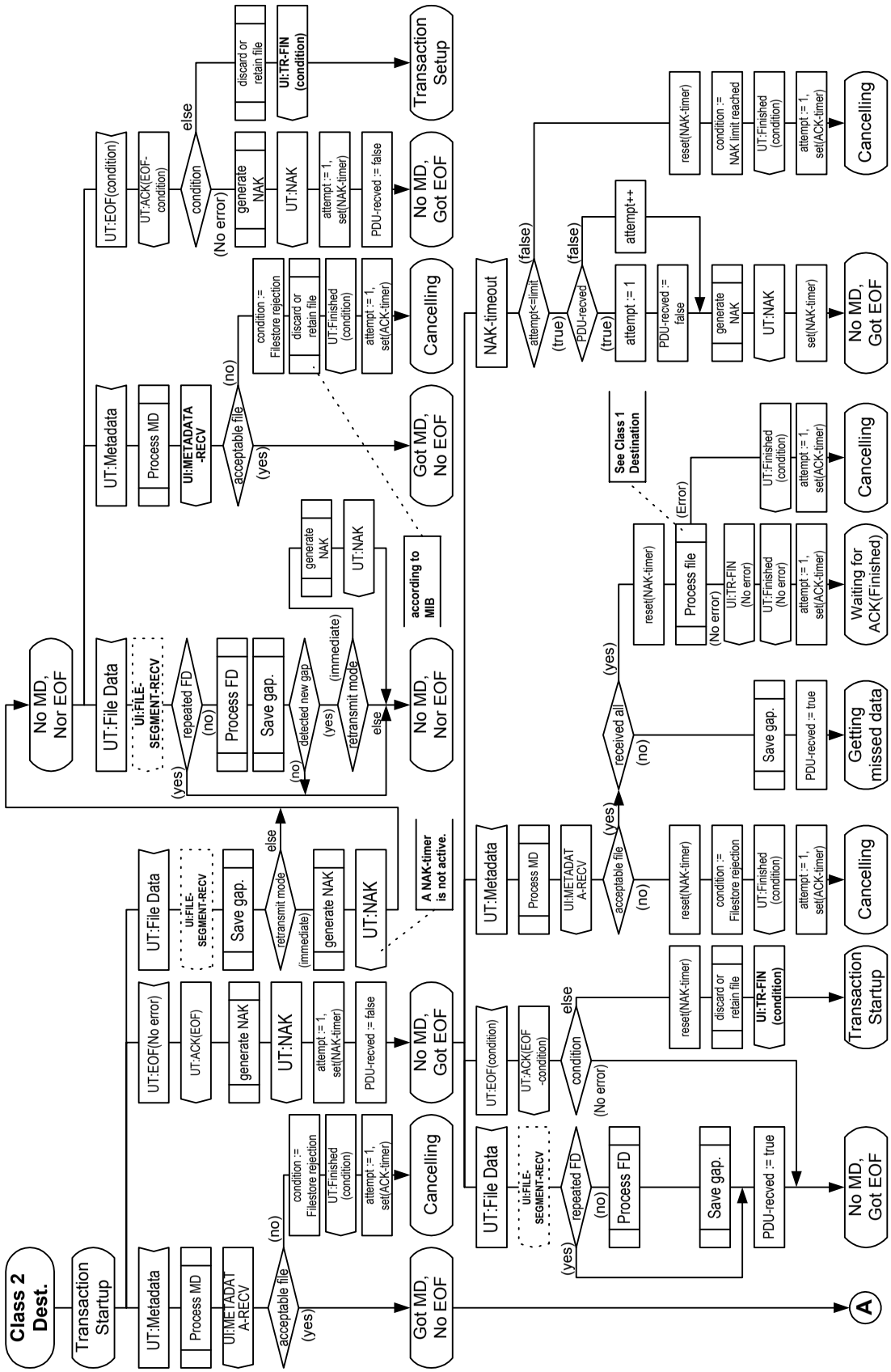


Figure 8-8: Class 2 Destination State Diagram

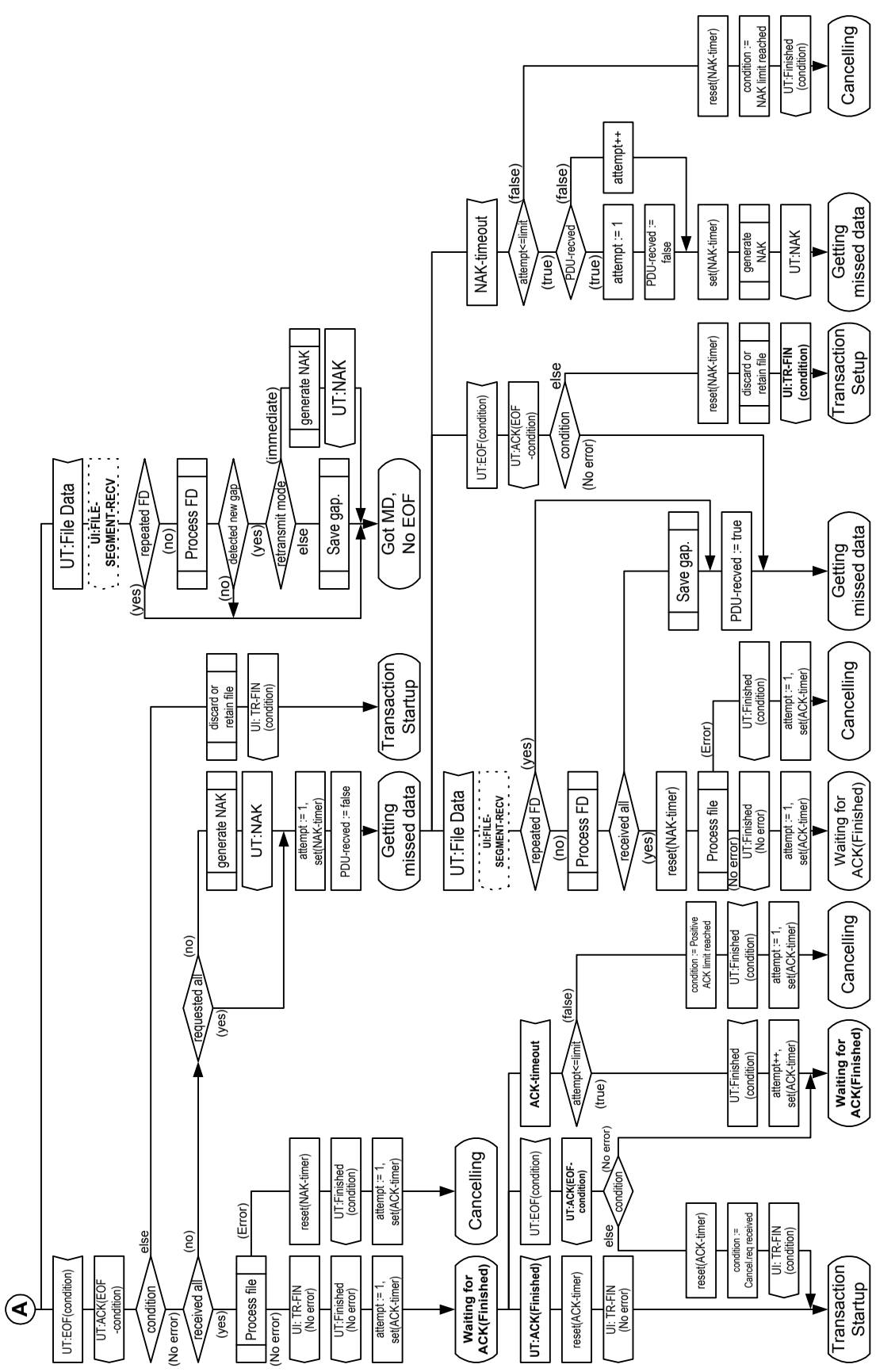


Figure 8-8: Class 2 Destination State Diagram (continued)

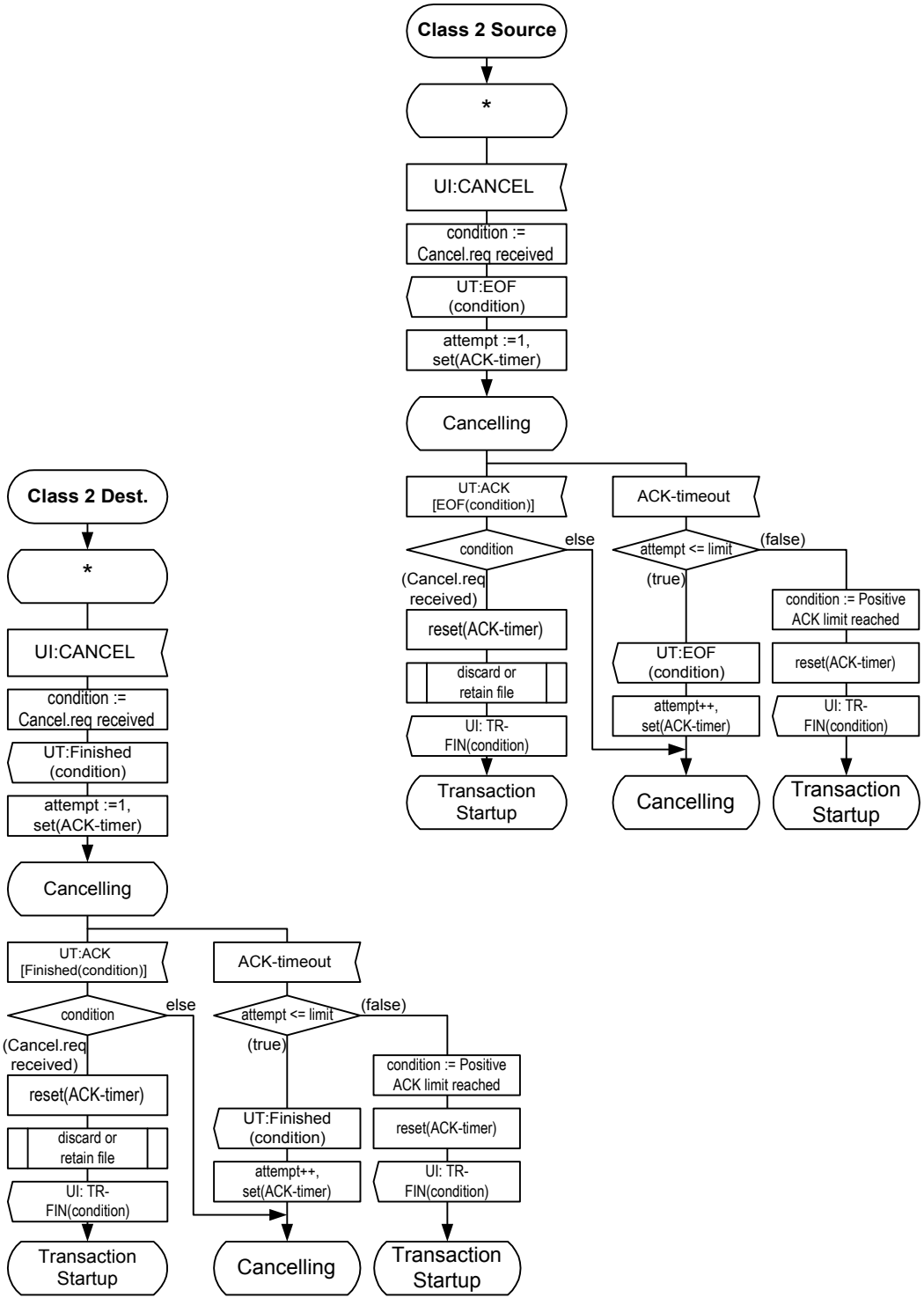


Figure 8-9: Class 2 Source or Destination State Diagram for Cancel

9 MANAGEMENT INFORMATION BASE

9.1 GENERAL

The operation of each CFDP entity shall be supported by a single Management Information Base (MIB) comprising the items of information described below.

NOTES

1 This section describes the Management Information Base.

2 Representation of, and mechanisms for access to, MIB items will be implementation matters. In particular, determination of which items will be static and which will be dynamic is a matter of implementation.

9.2 LOCAL ENTITY CONFIGURATION INFORMATION

For each item of local entity configuration information, a single value shall apply which shall pertain to the entire CFDP entity.

Table 9-1: Local Entity Configuration Information

Item	Comment
Local Entity ID	
`EOF-Sent.indication` required	True or false.
`File-Segment-Recv.indication` required	True or false.
`Transaction-Finished.indication` required when acting as receiving entity	True or false.
`Suspended.indication` required when acting as receiving entity	True or false.
`Resumed.indication` required when acting as receiving entity	True or false.
Default fault handlers	For each type of fault condition, a default handler (as enumerated in 4.1.7.2).

9.3 REMOTE ENTITY CONFIGURATION INFORMATION

For each item of remote entity configuration information, a single value shall apply for each remote entity with which the local CFDP entity may be in direct communication.

Table 9-2: Remote Entity Configuration Information

Item	Comment
Remote entity ID	
UT address	UT Address to use when transmitting to this entity.
One-way light time	For use in computing NAK and positive ACK timer intervals.
Total round-trip allowance for queuing delay	For use in computing NAK and positive ACK timer intervals.
Asynchronous NAK interval	Expressed as a time interval, or N/A.
Asynchronous Keep Alive interval	Expressed as a time interval, or N/A.
Asynchronous report interval	Expressed as a time interval, or N/A.
Immediate NAK mode enabled	True or false.
Prompt transmission interval	Expressed as a time interval, or N/A.
Default transmission mode	Acknowledged or unacknowledged.
Disposition of incomplete received file on transaction cancellation	Discard or retain.
CRCs required on transmission	True or false.
Maximum file segment length	In octets.
Keep Alive discrepancy limit	Expressed as a number of octets, or N/A.
Positive ACK timer expiration limit	Number of expirations.
NAK timer expiration limit	Number of expirations.
Transaction inactivity limit	A time limit.
Start of transmission opportunity	A signal produced by the operating environment.
End of transmission opportunity	A signal produced by the operating environment.
Start of reception opportunity	A signal produced by the operating environment.
End of reception opportunity	A signal produced by the operating environment.

NOTE – An implementation of CFDP may include a mechanism for automatically canceling, suspending, or abandoning a transaction whose total lifetime exceeds some predefined value. However, this behavior is not specified in the CFDP Protocol Specification and therefore no transaction lifetime limit is included in the definition of the MIB.

ANNEX A

INFORMATIVE REFERENCES

(This annex is **not** part of the Recommendation.)

[A1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-7. Yellow Book. Issue 7. Washington, D.C.: CCSDS, November 1996.

[A2] *Telecommand Summary of Concept and Rationale*. Report Concerning Space Data System Standards, CCSDS 200.0-G-6. Green Book. Issue 6. Washington, D.C.: CCSDS, January 1987.

[A3] *CCSDS File Delivery Protocol (CFDP)—Part 1: Introduction and Overview*. Draft Report Concerning Space Data System Standards, CCSDS 720.1-G-0.8. Draft Green Book. Issue 0.8. n.p.: n.p., July 2001.

[A4] *CCSDS File Delivery Protocol (CFDP)—Part 2: Implementers Guide*. Draft Report Concerning Space Data System Standards, CCSDS 720.2-G-0.9. Draft Green Book. Issue 0.9. n.p.: n.p., July 2001.

[A5] *Specification and Description Language (SDL)*. ITU Recommendation Z.100. Geneva: ITU, 1988.

ANNEX B

ACRONYMS

(This annex is **not** part of the Recommendation.)

Term	Meaning
ACK	Positive Acknowledgement
AOS	Advance Orbiting Systems
APID	Application Process Identifier
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
EOF	End of File
FDU	File Delivery Unit
LV	Length-Value
MIB	Management Information Base
MSB	Most Significant Bit
NAK	Negative Acknowledgement
NCC	Network Control Center
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
SAP	Service Access Point
SDL	Specification and Description Language
SDU	Service Data Unit
TC	Telecommand
TCP	Transmission Control Protocol
TLV	Type-Length-Value
TM	Telemetry
UDP	User Datagram Protocol
UT	Unitdata Transfer

ANNEX C

EXAMPLE OF CHECKSUM CALCULATION

(This annex is **not** part of the Recommendation.)

The following is an example of file checksum calculation, provided to illustrate the discussion in 4.1.5.1.1.8 d). In order to facilitate the discussion in this annex, the notes from page 4-4 are repeated here:

NOTE 1 – In order to include in a checksum the content of a file data PDU whose offset is not an integral multiple of 4, it is necessary to align the data properly before adding 4-octet blocks of it to the checksum. Data at offset Q may be aligned by inserting N octets of value 'zero' before the first octet of the data, where N = Q mod 4 (the remainder obtained upon dividing Q by 4).

NOTE 2 – In order to include in a checksum a sequence of M octets (the first of which is at a file offset that is an integral multiple of 4) where M is less than 4, it is necessary to pad the data to length 4 before adding it to the checksum. The data may be padded by inserting (4 – M) octets of value 'zero' after the last octet of the data.

Suppose we have a file that is 15 bytes long:

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e

Suppose, for illustrative purposes, that our maximum data segment size is 6. Then the file will be sent in three file data PDUs as follows:

Offset	Length	Content
0	6	00 01 02 03 04 05
6	6	06 07 08 09 0a 0b
12	3	0c 0d 0e

To compute the checksum, we divide the incoming file data into 4-octet words and sum them. We always divide at offsets that are integral multiples of 4. So the first word will be constructed by first copying the byte at offset 0 (which is an integral multiple of 4) into the high-order octet of the word:

00 __ __ __

and then copying the NEXT (not previous) three octets of file data into the next three octets of the word:

00 01 02 03

To construct the second word, we copy the byte at offset 4 into the high-order octet of the word:

04 __ __ __

and then copy the next three octets of file data into the next three octets of the word:

04 05 06 07

To construct the third word, we copy the byte at offset 8 into the high-order octet of the word:

08 __ __ __

and then copy the next three octets of file data into the next three octets of the word:

08 09 0a 0b

To construct the last word, where the remaining sequence of file data octets has length 3, we copy the byte at offset 12 into the high-order octet of the word:

0c __ __ __

then copy the remaining octets of file data into the next two octets of the word:

0c 0d 0e __

and finally (per NOTE 2) we insert (4 - 3) = 1 octets of value ‘zero’ after the last octet of the data:

0c 0d 0e 00

So we compute our checksum by adding these four 32-bit words:

00010203

+ 04050607

+ 08090a0b

+ 0c0d0e00

We carry in the usual way, except that anything that would be carried out of the high-order octet is simply discarded.

Now suppose we want to compute the checksum incrementally as each file data segment arrives, and suppose they arrive out of order:

Offset	Length	Content
0	6	00 01 02 03 04 05
12	3	0c 0d 0e
6	6	06 07 08 09 0a 0b

From the first file data segment we receive, we construct the following words:

00 01 02 03

04 05 00 00 (another application of NOTE 2)

From the second file data segment we receive, we construct the following words:

0c 0d 0e 00 (again, see NOTE 2)

From the third file data segment we receive, we construct words in a slightly different way, invoking the algorithm in NOTE 1. The offset of the file data PDU, 6, is not an integral multiple of 4, so we have to align the data before dividing it into words. To align data at offset Q we insert N preceding octets of value 'zero' before the first octet of the data, where N = Q modulo 4. Since 6 modulo 4 is 2, we have to insert two octets of value 'zero' before the first octet of file data. That changes

06 07 08 09 0a 0b

to

00 00 06 07 08 09 0a 0b

Now we construct the following words from this realigned file data:

00 00 06 07

08 09 0a 0b

So we compute our checksum by adding these words together:

00010203 (same as first word in previous case)

+ 04050000

+ 0c0d0e00 (same as fourth word in previous case)

+ 00000607

+ 08090a0b (same as third word in previous case)

Since 04050000 + 00000607 = 04050607, which was the second word in the previous case, we see that the order of arrival of the file data segment is immaterial to the incremental computation of the checksum.

ANNEX D

EXTENDED PROCEDURES

(This annex is **not** part of the Recommendation.)

D1 GENERAL

Extended procedures for conducting successive CFDP file copy operations among three or more entities in a store-and-forward manner have been drafted but have not been adopted by CCSDS. The sense of these procedures is documented here for future reference.

D2 RELEVANT AMENDMENTS TO SECTION 1

In 1.3.3.3:

Each end-to-end file transmission task includes one or more point-to-point file copy operations. In the simplest case, the only sender of the file is the source and the only receiver is the destination. In more complex cases (the general case) there are additional '*waypoint*' entities that receive and send copies of the file; the source is the first sender and the destination is the last receiver.

D3 RELEVANT AMENDMENTS TO SECTION 2

D3.1 In 2.1:

In its simplest form, the protocol provides a *Core* file delivery capability operating across a single link. For more complex mission scenarios, the protocol offers *Extended* operation providing store-and-forward functionality across an arbitrary network containing multiple links with disparate availability.

D3.2 In 2.2.2:

Each CFDP protocol entity has at most one (1) user. In some instances a user may not be present; in particular, any entity which always functions solely as a waypoint (performing store and forward operations) need not have a user.

D3.3 In 2.2.3:

The protocol entity consists of implementations of the *Core delivery procedures*, which allow immediate file delivery and manipulation over a single network hop, and optionally the *Extended procedures*, which allow for time-disjunct or immediate delivery over a number of network hops with appropriate facilities for onward routing. A single service interface is

presented to the user; the addition of the Extended procedures is evident in the quality of service and the multi-hop capability.

D3.4 In 2.3:

Where direct connectivity between the source and destination is impossible, the Extended procedures automatically execute multiple file copy operations: one file copy operation between the source and the first waypoint; others between successive waypoints as necessary; and a final file copy operation between the last waypoint and the destination. Each of these is simply another instance of the Core file copy operation.

D3.5 In 2.3.3:

The names, or 'entity IDs', that uniquely identify the source CFDP entity and the destination CFDP entity must be included in the header of each PDU in order for an FDU to be successfully transmitted through a series of waypoint CFDP entities.

D3.6 Additional subsection:

2.3.5 STORE AND FORWARD CONSIDERATIONS:

When the Extended procedures are implemented, the sender and/or receiver of a given PDU may be a 'waypoint' CFDP entity. Extended procedures are used when the original source of the PDU has no direct connectivity to the PDU's final destination, but only to some intermediate entity. That waypoint entity in turn may have direct connectivity either to the PDU's final destination or only to some further intermediate entity; the last waypoint entity in such a chain must have direct connectivity to the final destination of the PDU.

The determination of how and when a waypoint entity forwards a PDU toward its target entity is an implementation matter. In general it is desirable to forward each PDU as soon as possible, rather than wait until custody of an entire FDU has been taken before forwarding any part of it; this approach minimizes the time required for complete end-to-end transmission of the data. In practice, however, immediate forwarding will frequently be impossible, because radio contact among CFDP entities is typically discontinuous. The waypoint entity in such cases must store PDUs in some persistent medium, such as an intermediate copy of the transmitted file, until forwarding is practical.

The end-to-end execution of a transaction may therefore comprise multiple successive executions of Core procedures between adjacent entities, some of which may be initiated by the Extended procedures themselves rather than by a CFDP user's invocation of services; when this is the case, the Extended procedures essentially take on the role of the CFDP user. However, this variation is invisible to the Core procedures, which operate in the same way at all times.

When the sender for a Copy File procedure is the file's source entity but the receiver is a waypoint rather than the file's destination entity, the sender issues a `Transfer-Consigned.indication` primitive (rather than a `Transaction-Finished.indication` primitive)

when it receives notification from the receiver that the file has been successfully copied. This notifies the sending application that custody of the file has been transferred to the first waypoint; if the original file itself is being used as the sender's retransmission buffer, and is therefore protected from deletion or modification, it is now safe to end that protection.

When the receiver for a Copy File procedure is a waypoint, it may or may not wait until the entire procedure is complete before beginning to copy the file to the destination (or next waypoint). Immediate (incremental) forwarding of the file has the desirable effect of minimizing delay in getting at least part of the file to the destination.

The precise order in which stored PDUs are forwarded when connectivity is established is another implementation matter. Optional 'flow labels' may be associated with transmitted data to aid implementations of CFDP in determining how and when to issue the data.

D3.7 In 2.4:

It is possible for the source and destination CFDP entities of a transaction to be unable to communicate directly; in this case the transaction may entail a series of point-to-point (sender-to-receiver) PDU exchange sessions between the source and destination CFDP entities and one or more waypoint CFDP entities, as shown in figure 4-*n*.

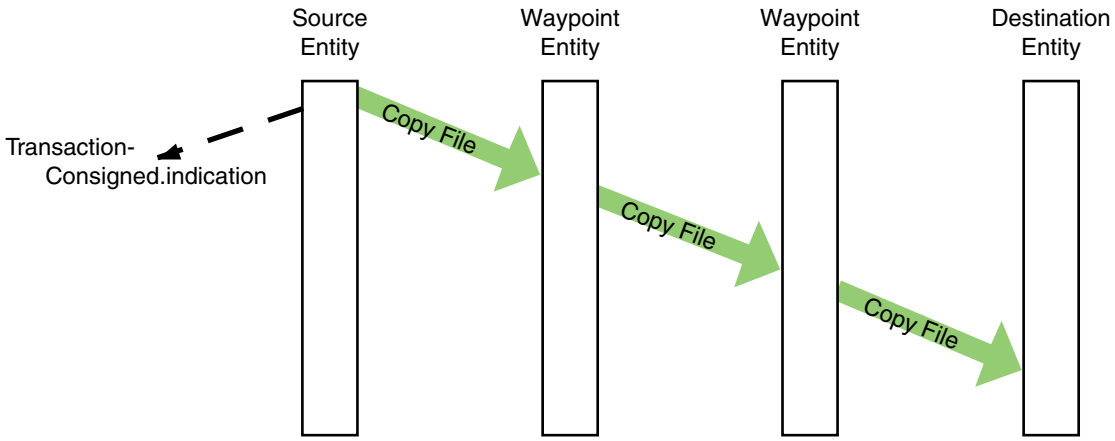


Figure 4-*n*: File Custody Transfer Operations, Sequence of Events

D4 RELEVANT AMENDMENTS TO SECTION 3

D4.1 Additional parageraph:

3.2.3 When the Extended procedures are operating, the CFDP service shall also deliver the following indication primitive: `Transfer-Consigned.indication`.

D4.2 Additional subsection:

3.5.15 `Transfer-Consigned.indication`

3.5.15.1 Function

When (and only when) the Extended procedures are in operation, the `Transfer-Consigned.indication` primitive shall be used to indicate, to the source CFDP user, the complete reception of a file by the first waypoint.

3.5.15.2 Semantics

`Transfer-Consigned.indication` shall provide parameters as follows:

`Transfer-Consigned.indication`	(transaction ID)

3.5.15.3 When Generated

`Transfer-Consigned.indication` shall be generated by the source entity on receipt of a Finished PDU which has a false End System Status flag.

3.5.15.4 Effect on Receipt

`Transfer-Consigned.indication` shall authorize the source CFDP user to modify or delete any retransmission buffer (file) that it was sharing with the protocol entity to conserve persistent storage space.

3.5.15.5 Additional Comments

The `Transfer-Consigned.indication` primitive is provided to support 'buffer sharing' in implementations for memory-constrained environments.

D5 RELEVANT AMENDMENTS TO SECTION 4

D5.1 In 4.1.4.3:

NOTE – When only the Core procedures are implemented, the sender and receiver of the file are the source and destination entities as identified in the PDU header. When the Extended procedures are implemented the senders and receivers of files must recognize themselves as such; the manner in which this is accomplished is an implementation matter.

D5.2 In 4.1.5, an additional NOTE:

3 When only the Core procedures are implemented, the sender of the file is always the transaction's source entity and the receiver of the file is always the transaction's destination entity.

D5.3 New paragraph:

4.1.5.4.1.5 The procedures for processing a Finished (No error) PDU with End System Status flag set to '0' are defined in the Extended procedures.

D5.4 In 4.1.5.4.2.5:

b) *if it is the destination CFDP entity*, issue a Finished (No error) PDU with End System Status flag set to '1'.

D5.5 In 4.1.5.6.1.3:

If the receiving entity is the destination entity, then the incomplete data shall be either discarded or retained according to the option set in the MIB.

The condition on which the procedures prescribed in 4.1.10.1.2.2 is amended to:

If the receiving entity is the destination entity,

D5.6 In 4.1.10.2.3.2:

If the receiving entity is the destination entity, then the incomplete data shall be either discarded or retained according to the option set in the MIB.

D5.7 In 4.1.10.3.2.3:

If the sending entity is the source entity, it shall issue a `Suspended.indication`.

D5.8 New subsection 4.2:

4.2 EXTENDED PROCEDURES

4.2.1 GENERAL

4.2.1.1 Because the Extended procedures supplement and rely on the capabilities provided by the Core procedures, a CFDP implementation that supports the Extended procedures shall also support all of the Core procedures without deviation from the specification.

4.2.1.2 In particular, waypoint entities involved in a Copy File procedure shall have the same ability to cancel, suspend, and resume the transaction as do the source and destination entities. To this end, whenever a CFDP entity is functioning as a waypoint (i.e., it is functioning as the receiving entity for a Copy File procedure whose file is destined for some other entity), the Extended procedures shall function in the capacity of the CFDP user: they shall receive all CFDP service indications issued in the course of the Copy File procedure and shall submit any necessary CFDP service requests.

4.2.1.3 For the purposes of the Extended procedures only, two general classes of CFDP entity shall be recognized:

a) an *atomic entity* is defined as a single instance of implementation of CFDP residing on a single processor;

b) an *aggregate entity* (or *aggregation*) is defined as a set of atomic entities that may be required to operate in concert as a single entity, for waypoint purposes only.

4.2.1.4 Each member of an aggregate entity shall be individually addressable and shall function as an atomic entity like any other, in addition to functioning in its role as a member of its aggregation, except that

a) no atomic entity may ever be a member of multiple aggregate entities, and

b) no member of an aggregate entity may ever act as an individual waypoint.

NOTE – This implies that an aggregate entity can act as a waypoint between one of its own member entities and some other entity, but no individual member of the aggregation can do so.

4.2.1.5 Each Copy File procedure shall involve exactly one sending entity and exactly one receiving entity, even if one or both of these entities are aggregate entities.

4.2.1.6 The nominal performance by an aggregate entity of any single element of a Core or Extended procedure shall in practice resolve to the performance of that procedure element by exactly one of the members of that aggregation.

4.2.1.7 An aggregation has no CFDP user and therefore cannot be either the source entity or the destination entity of a CFDP transaction. Otherwise the distinction between atomic and aggregate entities shall be invisible to the Core procedures, subject to the other General specifications of the Extended Procedures.

NOTES

1 In order to relay data between two entities residing in the same CFDP addressing domain, a waypoint entity must itself reside in the same CFDP addressing domain (that is, have access to the same underlying communication system). Therefore all members of a given aggregation necessarily reside in the same CFDP addressing domain, and all members of a given aggregation can use CFDP to communicate among themselves, either directly (through crosslinks) or indirectly (through other waypoint entities).

2 The order in which data are forwarded by the individual members of an aggregation – on multiple physical links – can never be assumed to be the order in which those data were originally transmitted, even when the UT layer guarantees in-sequence delivery on any single link. Features of CFDP that specifically rely on in-sequence data arrival should not be utilized in any operational scenario that includes forwarding through aggregate waypoints. These features include unacknowledged mode, the Keep Alive mechanism, and Incremental Lost Segment Detection.

3 No run-time mechanism for determining whether or not a given waypoint is an aggregation has yet been identified.

4.2.2 FILE DATA RELAY PROCEDURES

4.2.2.1 General

The following additional procedures shall apply whenever the receiving entity for a Copy File procedure is not the file's destination.

4.2.2.2 File Data Relay Procedures at the Receiving Entity in Unacknowledged Mode

Receipt of a Metadata, File Data, or EOF (No error) PDU shall cause the receiving CFDP entity to relay the PDU either to the destination entity or to another waypoint.

4.2.2.3 File Data Relay Procedures at the Sending Entity in Any Acknowledged Mode

On receipt of a Finished (No error) PDU with End System Status flag set to '0',

a) the sending CFDP entity shall issue a Notice of Consignment;

b) if the sending entity is a member of an aggregation, it shall send a Consigned handover message to every other member of the aggregation.

NOTE – The transaction remains unfinished after a Notice of Consignment and therefore remains liable to cancellation.

4.2.2.4 File Data Relay Procedures at the Receiving Entity in Any Acknowledged Mode

4.2.2.4.1 Following receipt of the first PDU for a new transaction, the receiving CFDP entity shall automatically initiate a second Copy File procedure, functioning as the sender.

NOTES

1 The receiver of the second Copy File procedure is either the file's destination or another waypoint. The receiving entity shall be selected at the time the procedure is initiated, based on the destination entity ID in the header of the PDU whose reception triggered initiation of the procedure.

2 The segmentation algorithm exercised by the waypoint entity in performing this Copy File procedure will not necessarily be the same as that exercised by any prior or subsequent sender of the file. That is, the sizes of File Data PDUs may be different during different Copy File procedures of the same transaction.

3 In the event that the waypoint entity is an aggregation, multiple members of the aggregation might physically receive different PDUs of the same transaction.

4 Features of CFDP that specifically rely on in-sequence data arrival should not be utilized in any operational scenario that relies on the Extended Procedures in any acknowledged mode. These features include the Keep Alive mechanism and Incremental Lost Segment Detection. In any acknowledged mode, data retransmission may cause the order in which file data segments are forwarded from a waypoint – even if the waypoint entity is not an aggregation – to differ from the order in which they were originally issued by the source entity.

4.2.2.4.2 Upon receipt of each Metadata PDU or File Data PDU, the recipient entity shall relay the PDU to the receiver of the second Copy File procedure.

4.2.2.4.3 When the Metadata, all file segments, and EOF (No error) PDU are deemed by the waypoint CFDP entity to have been received (that is, custody of the file has been acquired), the receiving CFDP entity shall relay the EOF (No error) PDU to the receiver of the second Copy File procedure and shall issue to the sending entity of the original Copy File procedure a Finished (No error) PDU with End System Status flag set to '0'.

4.2.2.4.4 In the event that the waypoint CFDP entity is an aggregation, the following specific File Data Relay Procedures shall apply:

a) physical receipt of a PDU by a member of the aggregation shall have the following additional effects:

   – if the PDU is a Metadata PDU, the aggregation member shall send a Metadata handover message to every other member of the aggregation,

- if the PDU is a File Data PDU, the aggregation member shall send a file data handover message to every other member of the aggregation,
- if the PDU is an EOF (No error) PDU, the aggregation member shall send an EOF handover message to every other member of the aggregation and the aggregation member shall be deemed 'cognizant' with respect to the referenced transaction;

b) recognition of collective reception of the metadata, all file data, and EOF (No error) for a given transaction by the cognizant member of the aggregation shall be deemed custody acquisition by the aggregate entity;

c) the cognizant member of the aggregation shall relay the EOF (No error) PDU to the receiver of the second Copy File procedure and shall issue to the sending entity of the original Copy File procedure a Finished (No error) PDU with End System Status flag set to '0' upon recognition of custody acquisition by the aggregate entity.

4.2.3 FINISHED (NO ERROR) PDU RELAY PROCEDURE

On receipt of any Finished (No error) PDU with End System Flag set to '1' at any CFDP entity that is not the source CFDP entity for the transaction, the following procedures shall apply in addition to the Acknowledged Mode Procedures at the Sending Entity defined in the Core Procedures:

a) the recipient CFDP entity shall relay the PDU either to the source entity or to another waypoint;

b) positive Acknowledgement procedures shall apply to the relayed Finished (No error) PDU;

c) in the event that the recipient CFDP entity is an aggregation, the aggregation member that physically receives the Finished (No error) PDU shall additionally send a Finished handover message to every other member of the aggregation.

4.2.4 CANCEL PROPAGATION PROCEDURES

4.2.4.1 On receipt of an EOF (cancel) PDU by a CFDP entity that is neither the source nor the destination of the file,

a) the entity shall issue a Notice of Completion (Canceled);

b) if an acknowledged mode is in effect, Positive Acknowledgement procedures shall be applied to the EOF (cancel) PDU with the Expected Response being an ACK (EOF) PDU;

c) the recipient CFDP entity shall relay the EOF (cancel) PDU toward the destination of the file;

d) if operating in acknowledged mode, Positive Acknowledgment procedures shall be applied to the relayed EOF (cancel) PDU with the Expected Response being an ACK (EOF) PDU;

e) if the recipient CFDP entity is a member of an aggregation, that entity shall additionally send a Cancel handover message citing the EOF (cancel) PDU's condition code to every other member of the aggregation.

4.2.4.2 On receipt of a Finished (cancel) PDU by a CFDP entity that is neither the source nor the destination of the file,

a) the entity shall issue a Notice of Completion (Canceled);

b) Positive Acknowledgement procedures shall be applied to the Finished (cancel) PDU with the Expected Response being an ACK (Finished) PDU;

c) the recipient CFDP entity shall relay the Finished (cancel) PDU toward the source of the file;

d) Positive Acknowledgment procedures shall be applied to the relayed Finished (cancel) PDU with the Expected Response being an ACK (Finished) PDU;

e) if the recipient CFDP entity is a member of an aggregation, that entity shall additionally send a Cancel handover message citing the Finished (cancel) PDU's condition code to every other member of the aggregation.

4.2.4.3 On receipt of a `Resume.request` primitive by a CFDP entity that is neither the source nor the destination of the file,

a) all of the core Resume Procedures shall apply, with the proviso that the recipient CFDP entity shall be considered both the Sending Entity and also the Receiving entity;

b) if the entity is a member of an aggregation, that entity shall additionally send a Resume handover message to every other member of the aggregation.

4.2.5 DEFERRED TRANSMISSION PROCEDURE

The CFDP entity shall retain for later transmission any PDU that cannot immediately be forwarded (because of link outage, for example) to the CFDP entity to which it is to be delivered or to the next waypoint, as applicable.

NOTES

1 The means by which the CFDP entity determines when PDUs can and cannot immediately be forwarded is an implementation matter. The CFDP entity might, for example, rely on additional service indications from the UT layer.

2 The flow label values associated with transactions may be taken as hints regarding the order in which PDUs should be transmitted when the opportunity arises, but the manner in which flow labels are used is strictly an implementation matter.

4.2.6 HANDOVER MESSAGE TRANSMISSION PROCEDURES

4.2.6.1 Handover messages shall be issued using a reliable protocol.

NOTE – Selection of the appropriate protocol is an implementation matter; options include TCP, SCPS-TP, and the Proximity-1 protocol with COP-1 enabled. However, interoperability among members of an aggregation is guaranteed only when CFDP itself is used to issue all handover messages.

4.2.6.2 When CFDP is used to issue a handover message,

a) the CFDP entity shall simulate the operation of a CFDP user in order to initiate a Put procedure;

b) each message shall occupy the Value of a single Handover Message TLV (defined in 5.4 below) in the metadata of the resulting FDU.

4.2.6.3 Receipt of any Metadata PDU containing one or more handover message TLVs shall cause the relevant handover message reception procedure to be performed for each such message.

4.2.7 HANDOVER MESSAGE RECEPTION PROCEDURES

4.2.7.1 Metadata Message Reception Procedure

Receipt of a Metadata message shall cause the (atomic) entity to recognize the collective reception of the affected transaction's metadata, by the aggregation of which the entity is a member.

4.2.7.2 File Data Message Reception Procedure

Receipt of a File Data Message shall cause the (atomic) entity to recognize collective reception of the indicated segment of the affected transaction's file data, by the aggregation of which the entity is a member.

4.2.7.3 EOF (No error) Message Reception Procedure

Receipt of an EOF (No error) message shall cause the (atomic) entity to recognize collective reception of the affected transaction's EOF (No error) PDU, by the aggregation of which the entity is a member.

4.2.7.4 Consigned Message Reception Procedure

Receipt of a Consigned message shall cause the (atomic) entity to release all unreleased portions of its file retransmission buffer for the affected transaction.

4.2.7.5 Finished Message Reception Procedure

Receipt of a Finished message shall cause the (atomic) entity to release all unreleased portions of its file retransmission buffer for the affected transaction and to stop transmission of file segments and metadata.

4.2.7.6 Suspend Message Reception Procedure

Receipt of a Suspend message shall cause the (atomic) entity to suspend all file delivery procedures for that transaction and save the status of the transaction.

4.2.7.7 Resume Message Reception Procedure

Receipt of a Resume message shall cause the (atomic) entity to resume all file delivery procedures for that transaction beginning at the file offset in the Resume message.

4.2.7.8 Cancel Message Reception Procedure

Receipt of a Cancel message shall cause the (atomic) entity to release all unreleased portions of its file retransmission buffer for the affected transaction, stop transmission of file segments and metadata, and stop requesting retransmission of file segments and metadata.

4.2.7.9 ACK Message Reception Procedure

Receipt of an ACK message shall cause the (atomic) entity to apply the Positive Acknowledgement procedures at the PDU Sending End, if and only if that entity physically sent the PDU whose acknowledgement is indicated by the message.

4.2.7.10 NAK Message Reception Procedure

Receipt of a NAK message shall cause the (atomic) entity to retransmit whatever data it has physically received that is requested by segment requests in the NAK message.

4.2.8 COLLECTIVE RESPONSE PROCEDURES

4.2.8.1 General

The following procedures shall apply whenever a PDU of the indicated type is physically received by an atomic entity that is a member of an aggregation.

4.2.8.2 ACK Response Procedure

On receipt of an ACK PDU,

- if the receiving entity physically sent the PDU whose acknowledgement is indicated by the message, that entity shall apply the Positive Acknowledgement procedures at the PDU Sending End;
- otherwise the entity shall send an ACK handover message (derived from this ACK PDU) to every other member of the aggregation.

4.2.8.3 NAK Response Procedure

4.2.8.3.1 Receipt of a NAK PDU shall cause the entity to retransmit whatever data it has physically received that is requested by segment requests in the NAK PDU.

4.2.8.3.2 If any data requested by the NAK PDU was not physically received by this entity, then the entity shall additionally send a NAK handover message (derived from this PDU) to every other member of the aggregation.

4.2.8.4 Prompt (NAK) Response Procedure

Receipt of a Prompt (NAK) PDU shall cause the entity to issue a NAK PDU requesting retransmission of all data not yet collectively received by the aggregation.

4.2.8.5 Prompt (Keep Alive) Response Procedure

Receipt of a Prompt (Keep Alive) PDU shall cause the entity to issue a Keep Alive PDU to the sending CFDP entity reporting on the transaction's collective reception progress so far at the aggregate entity.

4.2.8.6 Keep Alive Response Procedure

Receipt of a Keep Alive PDU shall optionally cause the entity to declare a Keep Alive Limit Reached fault if the discrepancy between the reception progress reported by the Keep Alive PDU and the transaction's collective transmission progress so far at the aggregate entity exceeds a preset limit.

4.2.9 COLLECTIVE KEEP ALIVE PROCEDURE

When Keep Alive procedures are active at an entity that is a member of an aggregation, each Keep Alive PDU issued by that entity shall report on the transaction's collective reception progress so far at the aggregate entity.

4.2.10 HANDOVER MESSAGES

4.2.10.1 General

4.2.10.1.1 Each handover message shall consist of a header in the following format, followed by message content whose format varies with the type of the message.

Field	Size (bits)	Values	Comment
Transmission mode	1	'0' – acknowledged '1' – unacknowledged	
Length of entity ID	3		Number of octets in entity ID less 1; i.e., '0' means that entity ID is 1 octet. Applies to all entity IDs in the handover message header.
Reserved	1	'0'	
Length of Transaction sequence number	3		Number of octets in sequence number less 1; i.e., '0' means that sequence number is 1 octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction to which the handover message pertains.
Transaction sequence number	variable		Uniquely identifies the transaction to which the handover message pertains, among all transactions originated by the source entity.
Destination entity ID	variable		Uniquely identifies the entity that is the final destination of the metadata and file data of the transaction to which the handover message pertains.
Message type	8	As defined in the following table	

4.2.10.1.2 Handover message types are defined as follows:

Type code (hexadecimal)	Type
00	File Data
01	(reserved)
02	(reserved)
03	(reserved)
04	EOF
05	Finished
06	ACK
07	Metadata
08	NAK
09	Consigned
0A	Suspend
0B	Resume
0C	Cancel
0D–FF	(reserved)

4.2.10.2 Metadata Message Content Field

The content of a Metadata message shall be as follows, with the values of all fields copied from the Metadata PDU whose reception caused the message to be issued:

Field	Size (bits)	Values	Comments
Segmentation Control	1	'0' - Record boundaries respected '1' - Record boundaries not respected	
Reserved for future use	7		Set to all 'zeroes'.
File size	32		In octets.

4.2.10.3 File Data Message Content Field

The content of a File Data Message shall be as follows:

Field	Size (bits)	Values	Comments
Segment Offset	32		In octets, copied from the segment offset of the File Data PDU whose reception caused the message to be issued.
Segment Length	32		In octets, the length of the file data in that PDU.

4.2.10.4 EOF Message Content Field

The content of an EOF Message shall be the content of an EOF (No error) PDU.

4.2.10.5 Consigned Message Content Field

The content of a Consigned Message shall be null.

4.2.10.6 Finished Message Content Field

The content of a Finished Message shall be null.

4.2.10.7 Cancel Message Content Field

The content of a Cancel Message shall be as follows:

Field	Size (bits)	Values	Comments
Condition Code	4	See table 5-5.	Copied from the condition code of the EOF (cancel) or Finished (cancel) PDU whose reception caused the message to be issued.
Spare	4		

4.2.10.8 ACK Message Content Field

The content of an ACK Message shall be as follows, with the values of all fields copied from the ACK PDU whose reception caused the message to be issued:

Field	Size (bits)	Values	Comments
Directive code	4	See table 5-4.	Directive code of the acknowledged PDU.
Directive subtype code	4	See table 5-8.	
Condition code	4	See table 5-5.	
Spare	2	All 'zeroes'.	
Transaction status	2	See table 5-8.	
Sender entity ID	LV		

4.2.10.9 NAK Message Content Field

The content of a NAK Message shall be the content of a NAK PDU.

4.2.11 INTERNAL EXTENDED PROCEDURES

4.2.11.1 Notice of Consignment Procedures

4.2.11.1.1 On Notice of Consignment of the Copy File procedure, the sending CFDP entity shall

a) release all unreleased portions of the file retransmission buffer;

b) stop transmission of file segments and metadata;

c) terminate any transmission of Prompt PDUs;

d) terminate the application of Positive Acknowledgment Procedures to PDUs previously issued by this entity.

4.2.11.1.2 If the sending entity is the source entity, the CFDP entity shall issue a `Transfer-Consigned.indication` primitive.

4.2.11.2 Notice of Cancellation Procedures

On Notice of Cancellation of the Copy File procedure at a CFDP entity that is neither the source nor the destination of the file,

a) all of the core Notice of Cancellation Procedures shall apply, with the proviso that the recipient CFDP entity shall be considered both the Sending Entity and also the Receiving entity;

   NOTE – The receiving CFDP entity for the resulting EOF (cancel) PDU is the file's destination or another waypoint. The receiving CFDP entity for the resulting Finished (cancel) PDU is the file's source or another waypoint.

b) if the entity is a member of an aggregation, that entity shall additionally send a Cancel handover message to every other member of the aggregation indicating the reason for transaction termination: `Cancel.request` received or the condition code of the fault whose declaration triggered the Notice of Cancellation.

4.2.11.3 Notice of Suspension Procedures

On Notice of Suspension of the Copy File procedure at a CFDP entity that is neither the source nor the destination of the file,

a) all of the core Notice of Suspension Procedures shall apply, with the proviso that the recipient CFDP entity shall be considered both the Sending Entity and also the Receiving entity;

b) if the entity is a member of an aggregation, that entity shall additionally send a Suspend handover message to every other member of the aggregation.

D6 RELEVANT AMENDMENTS TO SECTION 5

D6.1 In table 5-7, the additional End System Status value '0' is recognized. It signifies that the Finished PDU was generated by a waypoint.

D6.2 In table 5-8, additional values of Directive Subtype Code are recognized:

Values depend on file directive code. For ACK of Finished PDU: binary 0000 if the Finished PDU was generated by waypoint, binary 0001 if the Finished PDU was generated by end system.

D6.3 In table 5-7, the first "Reserved for future use" field is changed to "End-to-end transmission mode" with the following Comment: "If '0', waypoints must forward PDUs of this transaction using either Acknowledged mode or, alternatively, Unacknowledged mode with a reliable UT layer (such as TCP/IP or Proximity-1 with COP-P active). If '1', waypoints must forward PDUs of this transaction using Unacknowledged mode."

D6.4 Additional subsection:

5.4.6 HANDOVER MESSAGE TLV

The Type of the Handover Message TLV shall be 06 hex; the format of the Value shall be the format of a single Handover Message as defined above.

D7 RELEVANT AMENDMENTS TO SECTION 6

None.

D8 RELEVANT AMENDMENTS TO SECTION 7

D8.1 In 7.1, two additional classes are recognized:

d) Class 4—Unreliable Transfer Via One Waypoint;

e) Class 5—Reliable Transfer Via One Waypoint.

D8.2 Additional subsections 7.5 and 7.6:

7.5 FUNCTIONS OF CLASS 4—UNRELIABLE TRANSFER VIA ONE WAYPOINT

NOTE – Class 4 provides for the unreliable delivery of bounded or unbounded data files from a source to the destination through the mediation of a waypoint.

7.5.1 PROCEDURES FOR CLASS 4—SOURCE

For Class 4 functionality, the source CFDP entity shall use the procedures specified in table 7-8.

Table 7-8: Class 4 Source Procedures

Procedure	Subsection	Notes
Put Procedures	4.1.2	
Transaction Start Notification Procedure	4.1.3	
PDU Forwarding Procedures	4.1.4	Applicable to Metadata, File Data and EOF PDUs only.
Copy File Procedures at the Sending Entity	4.1.5.1.1	
Unacknowledged Mode Procedures at the Sending Entity	4.1.5.3.1	
Fault Handling Procedures	4.1.7	Only issuance of Notice of Cancellation is required.
Notice of Completion Procedures at the Sending Entity	4.1.10.1.1	
Notice of Cancellation Procedures at the Sending Entity	4.1.10.2.2	Positive acknowledgement not required.

7.5.2 PROCEDURES FOR CLASS 4—WAYPOINT

For Class 4 functionality, the waypoint CFDP entity shall use the procedures specified in table 7-9.

Table 7-9: Class 4 Waypoint Procedures

Procedure	Subsection	Notes
PDU Forwarding Procedures	4.1.4	Applicable to Metadata, File Data and EOF PDUs only.
Copy File Procedures at the Sending Entity	4.1.5.1.1	
Copy File Procedures at the Receiving Entity	4.1.5.1.2	
Unacknowledged Mode Procedures at the Sending Entity	4.1.5.3.1	
Unacknowledged Mode Procedures at the Receiving Entity	4.1.5.3.2	
Cancel Response Procedures at the Receiving Entity	4.1.5.6.1	Positive acknowledgement not required.
Fault Handling Procedures	4.1.7	Abandonment of transaction only required.
Notice of Cancellation Procedures	4.1.10.2	Positive acknowledgement not required.

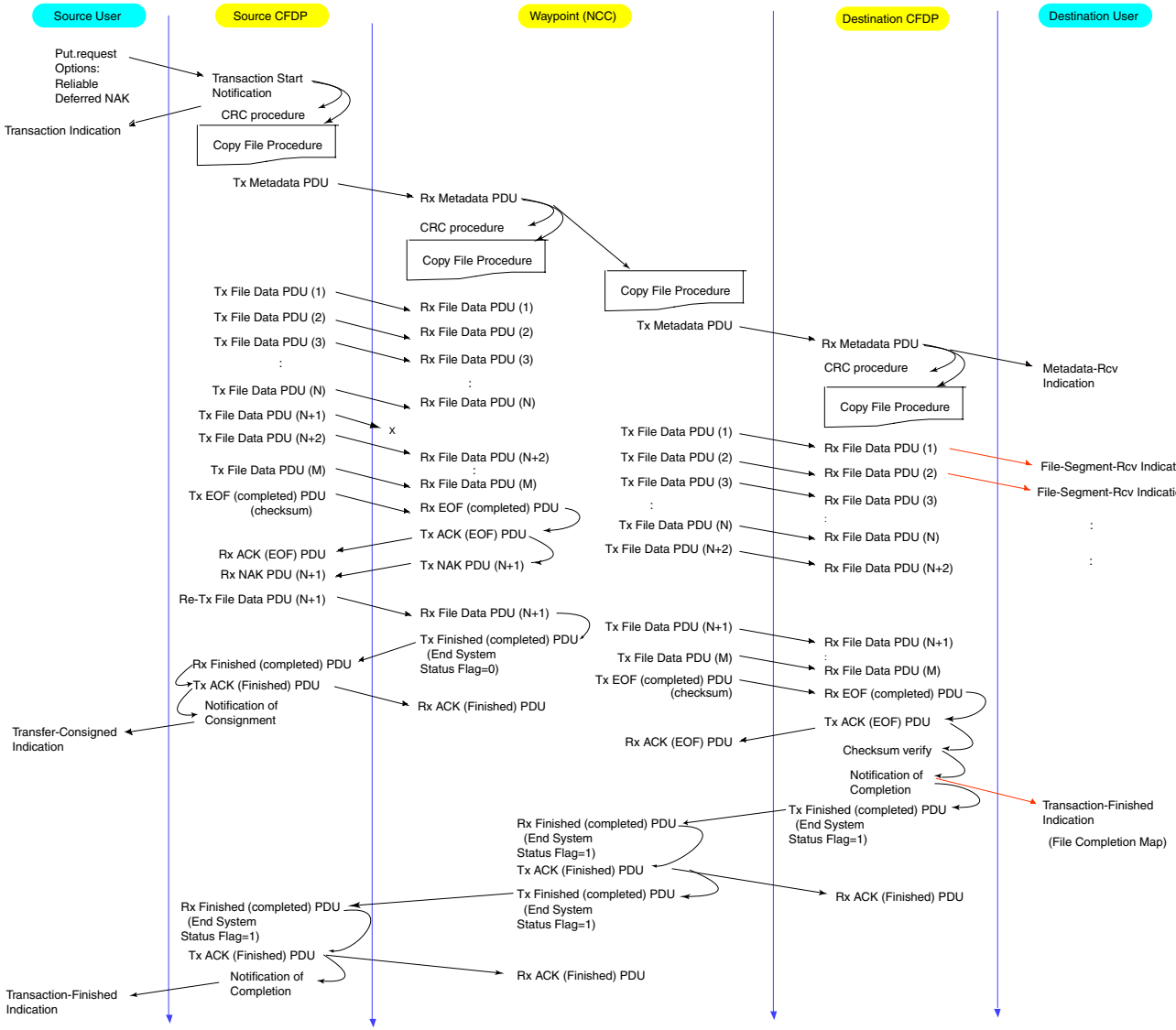
7.5.3 PROCEDURES FOR CLASS 4—DESTINATION

For Class 4 functionality, the destination CFDP entity shall use the procedures specified in table 7-10.

Table 7-10: Class 4 Destination Procedures

Procedure	Subsection	Notes
Copy File Procedures at the Receiving Entity	4.1.5.1.2	
Unacknowledged Mode Procedures at the Receiving Entity	4.1.5.3.2	
Cancel Response Procedures at the Receiving Entity	4.1.5.6.1	Positive acknowledgement not required.
Fault Handling Procedures	4.1.7	Abandonment of transaction only required.
Notice of Completion Procedures at the Receiving Entity	4.1.10.1.2	

7.5.4 EVENT DIAGRAM FOR CLASS 4



NOTE – Waypoint use of the Suspend/Resume Procedures and the Copy File Procedures at the Sending Entity entails potential interaction with a User Application. When a CFDP entity is operating as a Waypoint, the entity itself functions as the User Application; therefore for the sake of clarity in this diagram, interaction with the User Application is not shown. The potential interaction is limited to `EOF-Sent.indication`, `Suspend.indication`, and `Resume.request` as illustrated in the diagram for Service Class 2.

7.6 FUNCTIONS OF CLASS 5—RELIABLE TRANSFER VIA ONE WAYPOINT

NOTE – Class 5 provides for the reliable delivery of bounded or unbounded data files from a source to the destination through the mediation of a waypoint.

7.6.1 PROCEDURES FOR CLASS 5—SOURCE

For Class 5 functionality, the source CFDP entity shall use the procedures specified in table 7-11.

Table 7-11: Class 5 Source Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Transmitting Entity	4.1.1.1	
CRC Procedures at the PDU Receiving Entity	4.1.1.2	
Put Procedures	4.1.2	
Transaction Start Notification Procedure	4.1.3	
PDU Forwarding Procedures	4.1.4	
Copy File Procedures at the Sending Entity	4.1.5.1.1	
Acknowledged Mode Procedures at the Sending Entity	4.1.5.4.1	
Cancel Response Procedures at the Sending Entity	4.1.5.6.2	
Resume Procedures	4.1.5.7	Only procedures applicable to the sending entity.
Positive Acknowledgement Procedures at PDU Sending End	4.1.6.1	
Positive Acknowledgement Procedures at PDU Receiving End	4.1.6.2	
Fault Handling Procedures	4.1.7	Only issuance of Notice of Cancellation is required.
Internal Procedures	4.1.10	Positive acknowledgement required. Only procedures applicable to the sending entity are required.

7.6.2 PROCEDURES FOR CLASS 5—WAYPOINT

For Class 5 functionality, the waypoint CFDP entity shall use the procedures specified in table 7-12.

Table 7-12: Class 5 Waypoint Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Transmitting Entity	4.1.1.1	
CRC Procedures at the PDU Receiving Entity	4.1.1.2	
PDU Forwarding Procedures	4.1.4	
Copy File Procedures at the Sending Entity	4.1.5.1.1	
Copy File Procedures at the Receiving Entity	4.1.5.1.2	
Acknowledged Mode Procedures at the Sending Entity	4.1.5.4.1	
Acknowledged Mode Procedures at the Receiving Entity	4.1.5.4.2	
Incremental Lost Segment Detection Procedures at the Receiving Entity	4.1.5.4.3	Deferred Mode required only for interoperability.
Cancel Response Procedures	4.1.5.6	Positive acknowledgement required. Incomplete data forwarded.
Resume Procedures	4.1.5.7	
Positive Acknowledgement Procedures at PDU Sending End	4.1.6.1	
Positive Acknowledgement Procedures at PDU Receiving End	4.1.6.2	
Fault Handling Procedures	4.1.7	Only issuance of Notice of Cancellation is required.
Internal Procedures	4.1.10	
Finished (no error) PDU Relay Procedure	4.2.3	
Cancel Propagation Procedures	4.2.4	
Deferred Transmission Procedure	4.2.5	

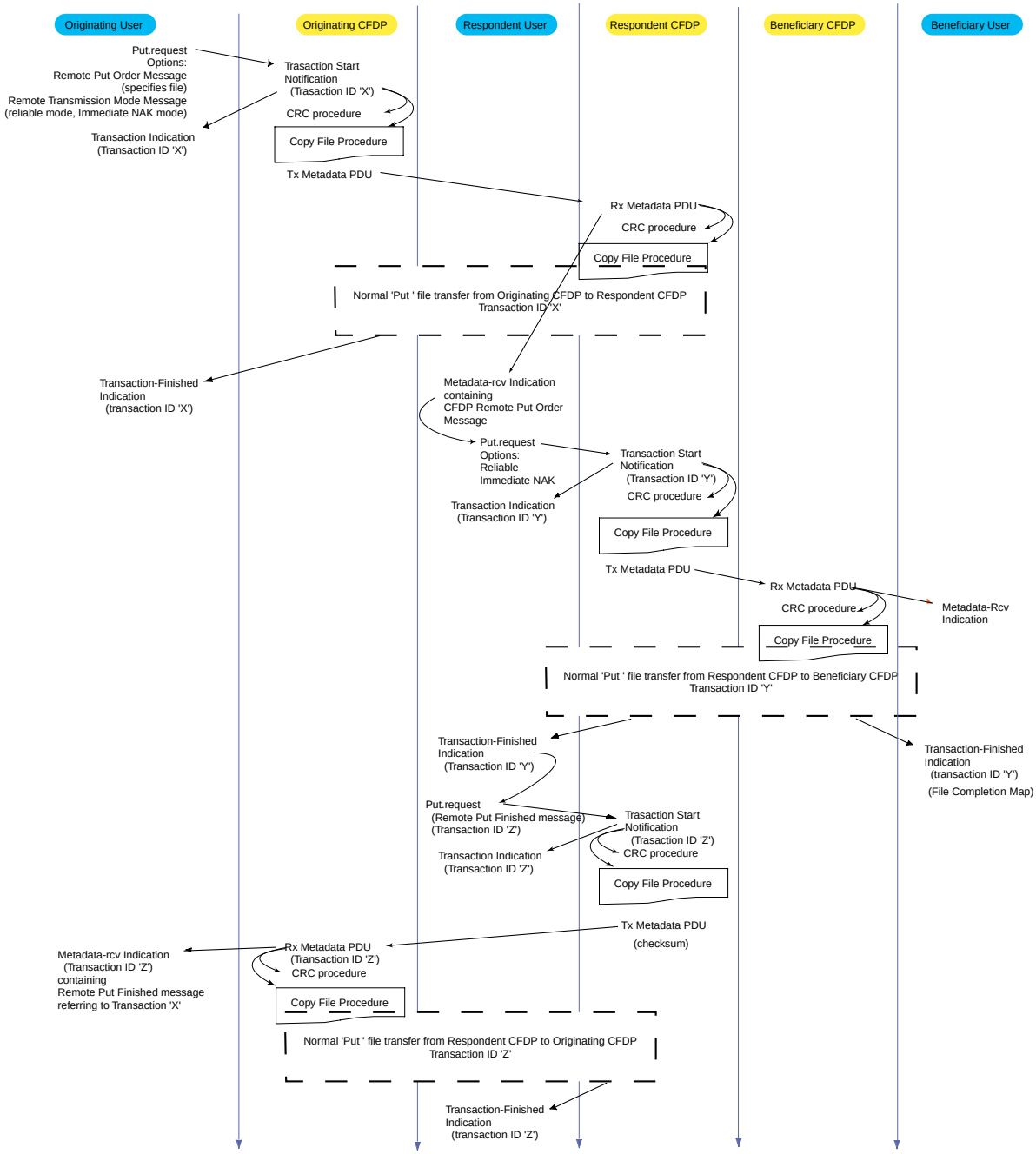
7.6.3 PROCEDURES FOR CLASS 5—DESTINATION

For Class 5 functionality, the destination entity shall use the procedures specified in table 7-13.

Table 7-13: Class 5 Destination Procedures

Procedure	Subsection	Notes
CRC Procedures at the PDU Transmitting Entity	4.1.1.1	
CRC Procedures at the PDU Receiving Entity	4.1.1.2	
PDU Forwarding Procedures	4.1.4	
Copy File Procedures at the Receiving Entity	4.1.5.1.2	
Acknowledged Mode Procedures at the Receiving Entity	4.1.5.4.2	
Incremental Lost Segment Detection Procedures at the Receiving Entity	4.1.5.4.3	Deferred Mode only required for interoperability.
Cancel Response Procedures at the Receiving Entity	4.1.5.6.1	Positive acknowledgement required. Incomplete data forwarded.
Resume Procedures	4.1.5.7	Only procedures applicable to the receiving entity are required.
Positive Acknowledgement Procedures at PDU Sending End	4.1.6.1	
Positive Acknowledgement Procedures at PDU Receiving End	4.1.6.2	
Fault Handling Procedures	4.1.7	Only issuance of Notice of Cancellation is required.
Internal Procedures	4.1.10	Incomplete data forwarded. Only procedures applicable to the receiving entity are required.
Deferred Transmission Procedure	4.2.5	

7.6.4 EVENT DIAGRAM FOR CLASS 5



NOTE – Waypoint use of the Suspend/Resume Procedures and the Copy File Procedures at the Sending Entity entails potential interaction with a User Application. When a CFDP entity is operating as a Waypoint, the entity itself functions as the User Application; therefore for the sake of clarity in this diagram, interaction with the User Application is not shown. The potential interaction is limited to EOF-Sent.indication, Suspend.indication, and Resume.request as illustrated in the diagram for Service Class 2.

D9 RELEVANT AMENDMENTS TO SECTION 8

D9.1 Additional state tables are defined:

Table 8-6: Unacknowledged Mode State Table, Waypoint

State: Event:	S0 Transaction Startup	S1 Waiting for EOF
E0 Entered this State (an implicit event whenever a state change occurs)	Initialize.	N/A
E12 Rx: Metadata PDU (This is normally the first event received)	Send this Metadata. State=S1	Send this Metadata.
E13 Rx: File-Data PDU	Issue File-Seg-Recv, Send this File-data. State=S1	Issue File-Seg-Recv, Send this File-data.
E14 Rx: EOF PDU	Send this EOF, Shutdown	Send this EOF, Shutdown
E23 Inactivity-timeout	N/A	Respond to inactivity timeout
E26 Rx: EOF-Cancel PDU	N/A	Send this EOF-Cancel, Shutdown
E36 Rx: Report Request	N/A	Issue Report

NOTE – All PDUs are received from upstream (i.e., from the Source or another waypoint) and transmitted downstream (i.e., to/towards the Destination).

Table 8-8: Acknowledged Mode State Table, Waypoint, Receiver Half

NOTE – This State Table is essentially a superset of the 'Acknowledged Mode, Receiver' State Table: States S7 & S8 are added. Events above E50 are added. The action for E17+S5 is different.

State: Event:	S0-S4 (same as for Acknowledged Mode, Receiver)	S5 Waiting for ACK-Finished	S6 Canceling	S7 Waiting for relay of Finished-Final	S8 Waiting for ACK-Finished
E0-E50 (except for E17 and E21)	Same as for Acknowledged Mode, Receiver	Same as for Acknowledged Mode, Receiver	Same as for Acknowledged Mode, Receiver	N/A	N/A
E17 Rx: ACK-Finished PDU	Same as for Acknowledged Mode, Receiver	State=S7	Same as for Acknowledged Mode, Receiver	N/A	Shutdown

E21
ACK-timer expired
	Same as for Acknowledged Mode, Receiver	Same as for Acknowledged Mode, Receiver	Same as for Acknowledged Mode, Receiver	N/A	If (Attempts < Max_Attempts)
					Resend Finished-Final,
					Increment Attempts,
					Restart ACK-timer
					Else
					Fault

E55 Please relay this Finished-Final	N/A	N/A	N/A	Send this Finished-Final, Set Attempts=1, Start ACK-timer State=S8	N/A
E61 Waypoint-partner Canceled	TBD	TBD	TBD	TBD	TBD
E62 Waypoint-partner Suspended	TBD	TBD	TBD	TBD	TBD
E63 Waypoint-partner Resumed	TBD	TBD	TBD	TBD	TBD

Table 8-9: Acknowledged Mode State Table, Waypoint, Sender Half

State: Event:	S0 Transaction Startup	S1 Waiting for relay of EOF	S2 Waiting for ACK-EOF	S3 Waiting for Finished	S4 Waiting for Finished-final	S5 Canceling
E0 Entered this state (implicit)	Initialize.	N/A	N/A	N/A	N/A	N/A
E15 Rx: ACK-EOF PDU	N/A	N/A	Stop ACK-timer State=S3	N/A	N/A	N/A
E16 Rx: Finished-final PDU	N/A	N/A	N/A	Send ACK-Finished, Shutdown	Send ACK-Finished, Shutdown	N/A
E17 Rx: ACK-Finished PDU	N/A	N/A	N/A	N/A	N/A	Shutdown

E18
Rx: NAK PDU
	N/A	If (Does NAK include Metadata)
		If (Has_MD_arrived)
		If (Attempts < Max_attempts)
		Resend Metadata,
		Increment Attempts
		Else
		Fault
		If (Does NAK include File-data)
		Resend file-data if available
			If (Does NAK include Metadata)
			If (Has_MD_arrived)
			If (Attempts < Max_attempts)
			Resend Metadata,
			Increment Attempts
			Else
			Fault
			If (Does NAK include File-data)
			Resend file-data if available
				If (Does NAK include Metadata)
				If (Has_MD_arrived)
				If (Attempts < Max_attempts)
				Resend Metadata,
				Increment Attempts
				Else
				Fault
				If (Does NAK include File-data)
				Resend file-data if available
					NA	N/A

E21
ACK-timeout
	N/A	N/A	If (attempts < Max_Attempts)
			Resend EOF,
			Increment Attempts,
			Start ACK-timer
			Else
			Fault
				N/A	N/A	If (attempts < Max_Attempts)
						Resend EOF-Cancel,
						Increment Attempts,
						Start ACK-timer
						Else
						Shutdown

Table 8-9: Acknowledged Mode State Table, Waypoint, Sender Half (continued)

State: Event:	S0 Transaction Startup	S1 Waiting for relay of EOF	S2 Waiting for ACK-EOF	S3 Waiting for Finished	S4 Waiting for Finished-final	S5 Canceling
E23 Inactivity Timeout	N/A	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	Respond to Inactivity Timeout	N/A
E28 Rx: Finished-Cancel PDU	N/A	Send ACK-Finished, Shutdown	Send ACK-Finished, Shutdown	Send ACK-Finished, Shutdown	Send ACK-Finished, Shutdown	Send ACK-Finished, Shutdown
E29 Rx: Cancel Request	N/A	Send EOF-Cancel, Set Attempts=1, Start ACK-timer, Trigger E49 in Waypoint-partner, State=S5	N/A	N/A	N/A	N/A
E31 Rx: Suspend Request	N/A	TBD	TBD	TBD	TBD	N/A
E34 Rx: Resume Request	N/A	TBD	TBD	TBD	TBD	N/A
E36 Rx: Report Request	N/A	Issue report	Issue report	Issue report	Issue report	Issue report
E42 Please send Prompt-NAK	N/A	Send Prompt-NAK	Send Prompt-NAK	Send Prompt-NAK	N/A	N/A
E44 Please send Prompt-Keep Alive	N/A	Send Prompt-Keep Alive	Send Prompt-Keep Alive	Send Prompt-Keep Alive	N/A	N/A
E46 Rx: Keep Alive PDU	N/A	If (!Is Keep Alive offset valid) Fault	If (!Is Keep Alive offset valid) Fault	N/A	N/A	N/A
E51 Rx: Finished-waypoint PDU	N/A	N/A	N/A	Send ACK-Finished, State=S4	Send ACK-Finished	N/A
E52 Please relay this Metadata PDU	Has_MD_arrived=Yes, Send Metadata, State=S1	Has_MD_arrived=Yes, Send Metadata	Has_MD_arrived=Yes, Send Metadata	Has_MD_arrived=Yes, Send Metadata	N/A	N/A

Table 8-9: Acknowledged Mode State Table, Waypoint, Sender Half (continued)

State: Event:	S0 Transaction Startup	S1 Waiting for relay of EOF	S2 Waiting for ACK-EOF	S3 Waiting for Finished	S4 Waiting for Finished-final	S5 Canceling
E53 Please relay this File-data PDU	Store File-data, Send File-data	Store File-data, Send File-data	Store File-data, Send File-data	Store File-data, Send File-data	N/A	N/A
E54 Please relay this EOF PDU	Send EOF, Set Attempts=1, Start ACK-timer, State=S2	Send EOF, Set Attempts=1, Start ACK-timer, State=S2	N/A	N/A	N/A	N/A
E56 Please relay this EOF-Cancel PDU	Send EOF-Cancel, Set Attempts=1, Start ACK-timer, State=S5	Send EOF-Cancel, Set Attempts=1, Start ACK-timer, State=S5	N/A	N/A	N/A	N/A
E61 Waypoint-partner Canceled	N/A	Send EOF-Cancel, Set Attempts=1, Start ACK-timer, State=S5	N/A	N/A	N/A	N/A
E62 Waypoint-partner Suspended	N/A	TBD	TBD	TBD	TBD	N/A
E63 Waypoint-partner Resumed	N/A	TBD	TBD	TBD	TBD	N/A

D9.2 Also, additional events are defined.

8.2.4.9 Service Class 4 Waypoint Support

E51 – Received a Finished PDU from a waypoint (see 'Finished-Cancel').

NOTE – The following set of events covers situations in which PDUs must be relayed through the waypoint (e.g., the Waypoint-Receiver receives a Metadata PDU, and the Waypoint-Sender must relay it to/toward the final destination):

E52 – Metadata PDU for relay downstream.

E53 – File-data PDU for relay downstream.

E54 – EOF PDU for relay downstream.

E55 – Finished PDU from final destination for relay upstream.

E56 – EOF-Cancel PDU for relay downstream.

E57 – Finished-Cancel PDU for relay upstream.

E60 – Reserved for future expansion.

NOTE – When an action (e.g., canceling a transaction) is initiated *within* a waypoint, PDUs must be sent in both directions (towards both the original Sender and the final Receiver). The following events support that need.

E61 – Waypoint-partner Canceled.

E62 – Waypoint-partner Suspended.

E63 – Waypoint-partner Resumed.

D10 RELEVANT AMENDMENTS TO SECTION 9

Additional item of local entity configuration MIB information:

Routing information (Extended procedures only)	For each entity in the system, a list of all adjacent entities.